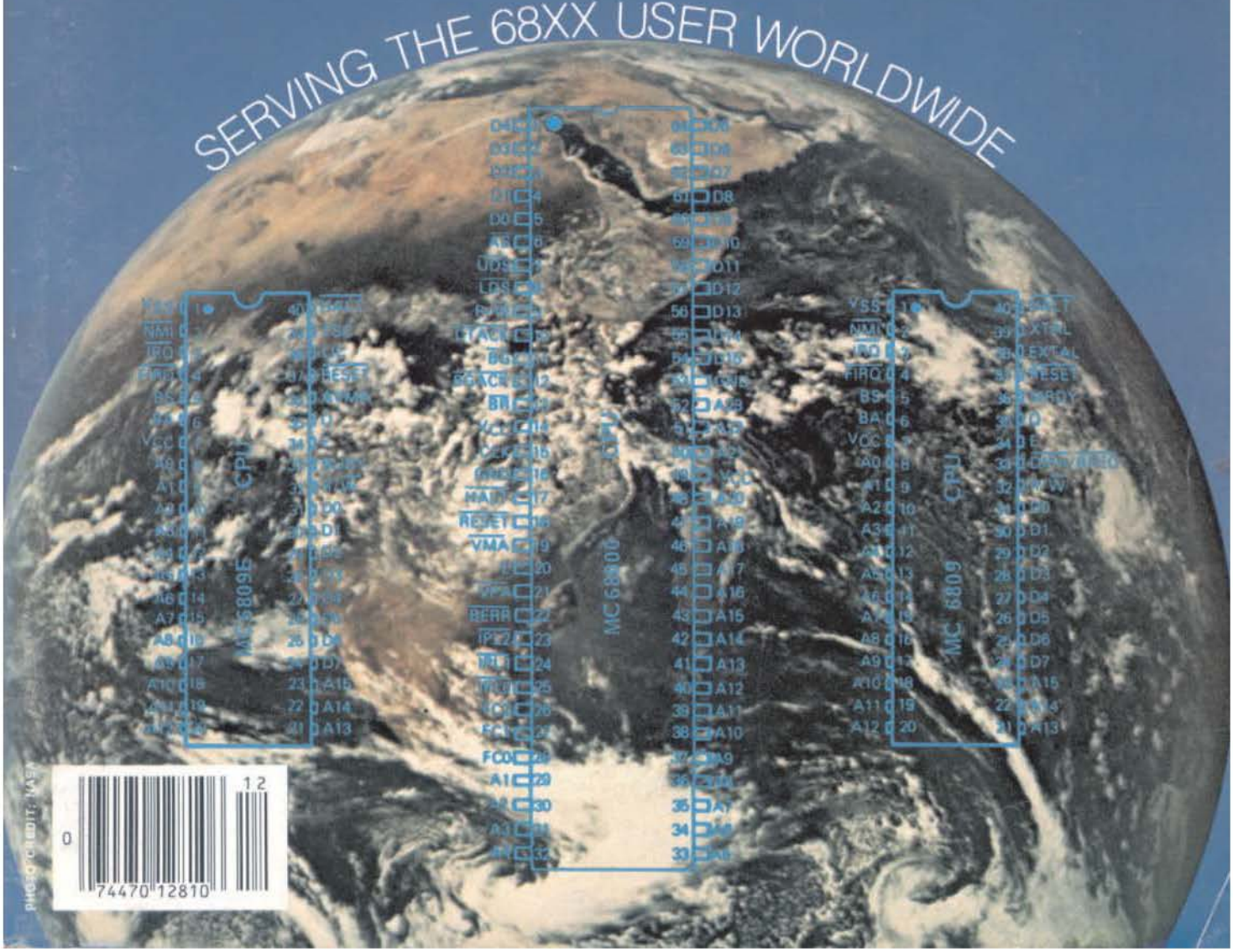
'68' MICRO JOURNAL

$2.95 USA

Australia	A $4.75	New Zealand	NZ $ 6.50
Singapore	S $ 9.45	Hong Kong	H $23.50
Malaysia	M $ 9.45	Sweden	30 SEK

VOLUME VI ISSUE XII • Devoted to the 68XX User • December 1984

"Small Computers Doing Big Things"



12
0 74470 12810

WE DON'T PLAY GAMES

X-12+ A SERIOUS COMPUTER IN A DESKTOP PACKAGE

Multiprocessor Technology - Combination of 8, 16 and 32 bit types

1.0 Megabyte Memory - Insures no limitation on programs

"Winchester" Disk System - Fast response, large storage capacity

UniFlex* Operating System - The standard of comparison

Hardware Floating Point - Unmatched speed in a small system

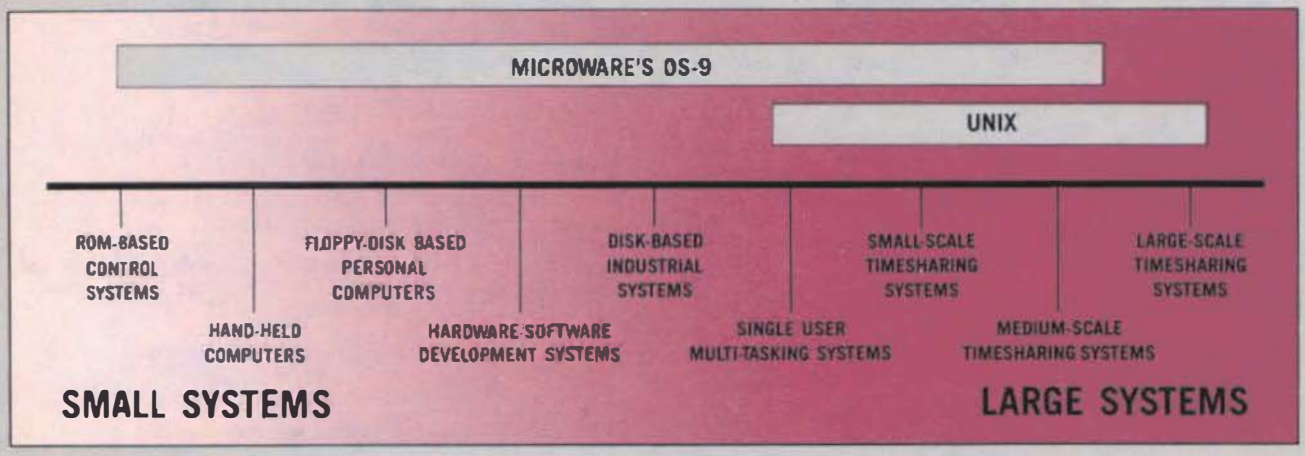
Up to Three Terminals - Instant expansion

* Trademark of Technical Systems Consultants

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

Only Microware's OS-9 Operating System Covers the Entire 68000 Spectrum



Is complicated software and expensive hardware keeping you back from Unix? Look into OS-9, the operating system from Microware that gives 68000 systems a Unix-style environment with much less overhead and complexity.

OS-9 is versatile, inexpensive, and delivers outstanding performance on any size system. The OS-9 executive is much smaller and far more efficient than Unix because it's written in fast, compact assembly language, making it ideal for critical real-time applications. OS-9 can run on a broad range of 8 to 32 bit systems based on the 68000 or 6809 family MPUs from ROM-based industrial controllers up to large multiuser systems.

Key OS-9 Features At A Glance

- Compact (16K) ROMable executive written in assembly language
- User "shell" and complete utility set written in C
- C-source code level compatibility with Unix
- Full Multitasking/multiuser capabilities
- Modular design - extremely easy to adapt, modify, or expand
- Unix-type tree structured file system
- Rugged "crash-proof" file structure with record locking
- Works well with floppy disk or ROM-based systems
- Uses hardware or software memory management
- High performance C, Pascal, Basic and Cobol compilers

OS-9'S OUTSTANDING C COMPILER IS YOUR BRIDGE TO UNIX

Microware's C compiler technology is another OS-9 advantage. The compiler produces extremely fast, compact, and ROMable code. You can easily develop and port system or application software back and forth to standard Unix systems. Cross-compiler versions for VAX and PDP-11 make coordinated Unix/OS-9 software development a pleasure.

SUPPORT FOR MODULAR SOFTWARE — AN OS-9 EXCLUSIVE

Comprehensive support for modular software puts OS-9 a generation ahead of other operating systems. It multiplies programmer productivity and memory efficiency. Application software can be built from individually testable software modules including standard "library" modules. The modular structure lets you customize and reconfigure OS-9 for specific hardware easily and quickly.

A SYSTEM WITH A PROVEN TRACK RECORD

Once an underground classic, OS-9 is now a solid hit. Since 1980 OS-9 has been ported to over a hundred 6809 and 68000 systems under license to some of the biggest names in the business. OS-9 has been imbedded in numerous consumer, industrial, and OEM products, and is supported by many independent software suppliers.

microware® OS-9™

MICROWARE SYSTEMS CORPORATION
1866 NW 114th Street
Des Moines, Iowa 50322
Phone 515-224-1929
Telex 910-520-2535

Microware Japan, Ltd
3-8-9 Baraki, Ichikawa City
Chiba 272-01, Japan
Phone 0473(28)4493
Telex 299-3122

'68' MICRO JOURNAL

Portions of the text for 68 MICRO JOURNAL was prepared using the following furnished hard/software.

COMPUTERS-HARDWARE
Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216
S09-5/8 DMF disk-CDS1-8212W-Sprint 3 Printer

GIMIX Inc.
1337 West 37th Place
Chicago, IL 60609
Super Mainframe-OS9-FLEX-Assorted Hardware

EDITORS-WORD PROCESSORS
Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, NC 27514
FLEX-Editor-Processor

Great Plains Computer Company, Inc.
PO Box 916
Idaho Falls, ID 83401
STYLO-Mail Merge

CONTENTS

Vol.VI, Issue XII — December 84

Send All Correspondence To:

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
Ph (615)842-4600 TELEX 558 414 PVT BTH

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, Tenn. and additional entries. Postmaster: send Form 3579 to 68' Micro Journal, PO Box 849, Hixson, Tennessee.

SUBSCRIPTION RATES

USA

1-Year $24.50 2-Years $42.50 3-Years $64.50

FOREIGN

See Page 60

Items Submitted for Publication

Articles submitted for publication should be accompanied by the authors **full name, address, date and telephone number.** It is preferred that articles be submitted on either 5 or 8 inch diskette in TSC Editor format or STYLO format. All diskettes will be returned.

The following TSC Text Processor commands **ONLY** should be used (due to our proportional processor): .sp space, .pp paragraph, .fi fill and .nf no fill. Also please do not format within the text with multiple spaces. The rest we will enter at time of editing.

STYLO commands are all acceptable except the .pg page command, we print edited text files in continous text.

All articles submitted on diskettes should be in TSC FLEX™ format, either FLEX2 6800, or FLEX9 6809 any version.

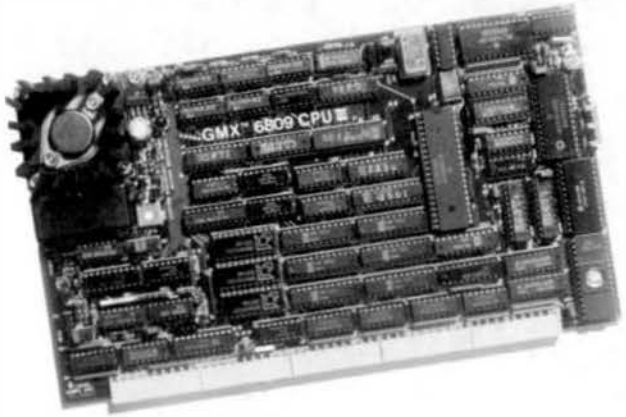
If articles are submitted on paper they should be on white 8X11 bond or better grade paper. No hand written articles (hand written or drawn art accepted). All paper submitted articles will be photo reproduced. This requires that they be typed or produced with a dark ribbon (no blue), single spaced and type font no smaller than 'elite' or 12 pitch. Typed text should be approximately 7 inches wide (will be reduced to column width of 3 1/2 inches). **Please use a dark ribbon!**

All letters to the editor should also comply with the above and bear a signature. Letters of 'gripes' as well as 'praise' are solicited. We attempt to publish all letters to the editor verbatim, however, we reserve the right to reject any submission for lack of 'good taste'. We reserve the right to define what constitutes 'good taste'.

Advertising: Commercial advertisers please contact the 68 Micro Journal advertising department for current rate sheet and requirements.

Classified: All classified must be non-commercial. Maximum 20 words per classified ad. Those consisting of more than 20 words should be figured at .35 cents per word. 20 words or less $7.50 minimum, one time, paid in advance. No classified ads accepted by telephone.

GIMIX STATE OF THE ART 6809 SYSTEMS FOR THE SERIOUS USER.



GIMIX has 19MB or high performance 47MB Winchester Drive Systems and/or Floppy Disk Drive Systems.

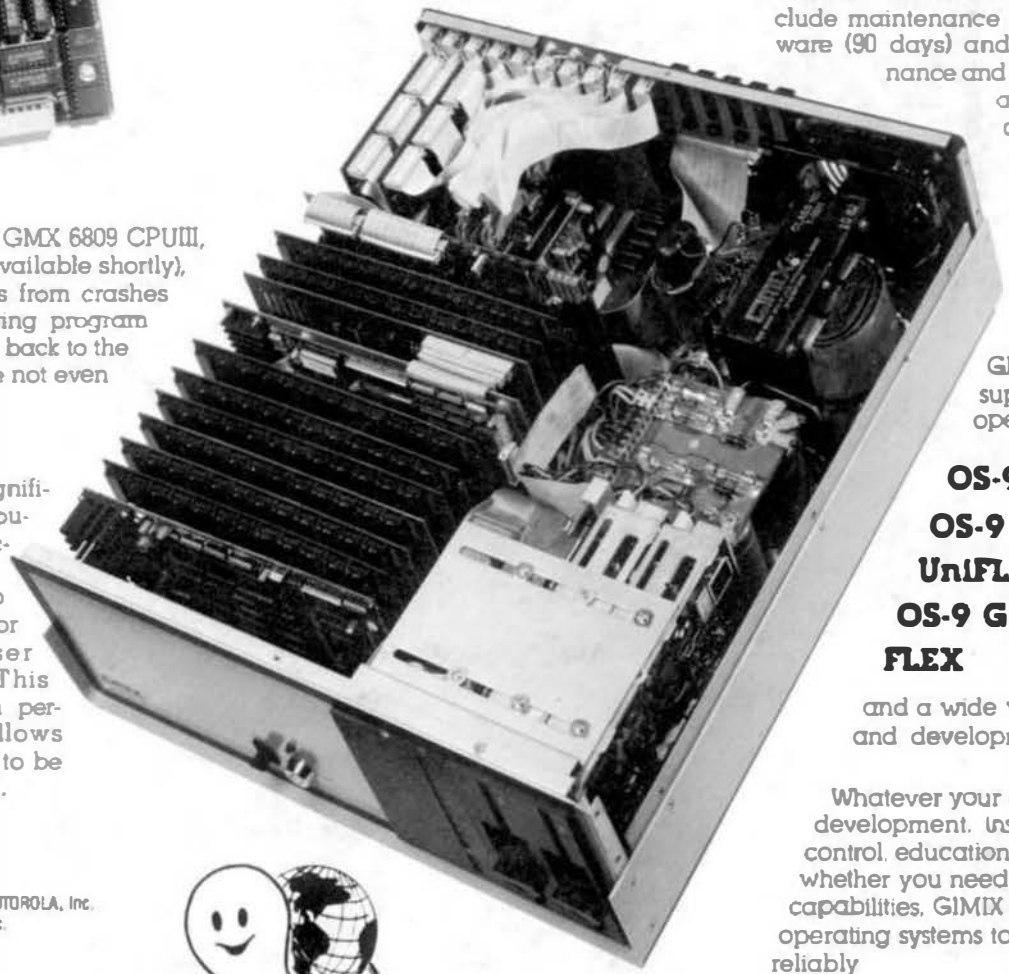
For the user who appreciates the need for a bus structured system using STATIC RAM and powered by a ferro resonant constant voltage transformer.

GIMIX has single user systems that can run both FLEX and OS-9 or Multi user systems for use with UniFLEX or OS-9.

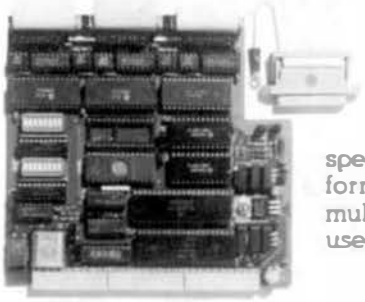
GIMIX versions of OS9 and UniFLEX include maintenance and support by Microware (90 days) and TSC (1 year). Maintenance and support after this period are available at extra cost.

(NOTE: this support and maintenance is only for use with approved GIMIX hardware)

For the ultimate in performance, the Unique GMX 6809 CPUIII, using either OS-9-GMXIII or UniFLEX GMXIII (available shortly), gives protection to the system and other users from crashes caused by defective user programs. e.g. During program development, a programmer who crashes goes back to the shell or the debugger, while the other users are not even aware anything occurred.



The intelligent serial I/O processor boards significantly reduce system overhead by handling routine I/O functions, thereby freeing up the host CPU for running user programs. This speeds up system performance and allows multiple terminals to be used at 19.2K baud.



GIMIX 6809 systems support five predominant operating systems:

OS-9 GMX III,
OS-9 GMX II,
UniFLEX,
OS-9 GMX I,
FLEX

and a wide variety of languages and development software.

Whatever your application: software development, instrumentation, process control, educational, scientific or business; whether you need single or multi-user capabilities, GIMIX has hardware and the operating systems to get the job done reliably.

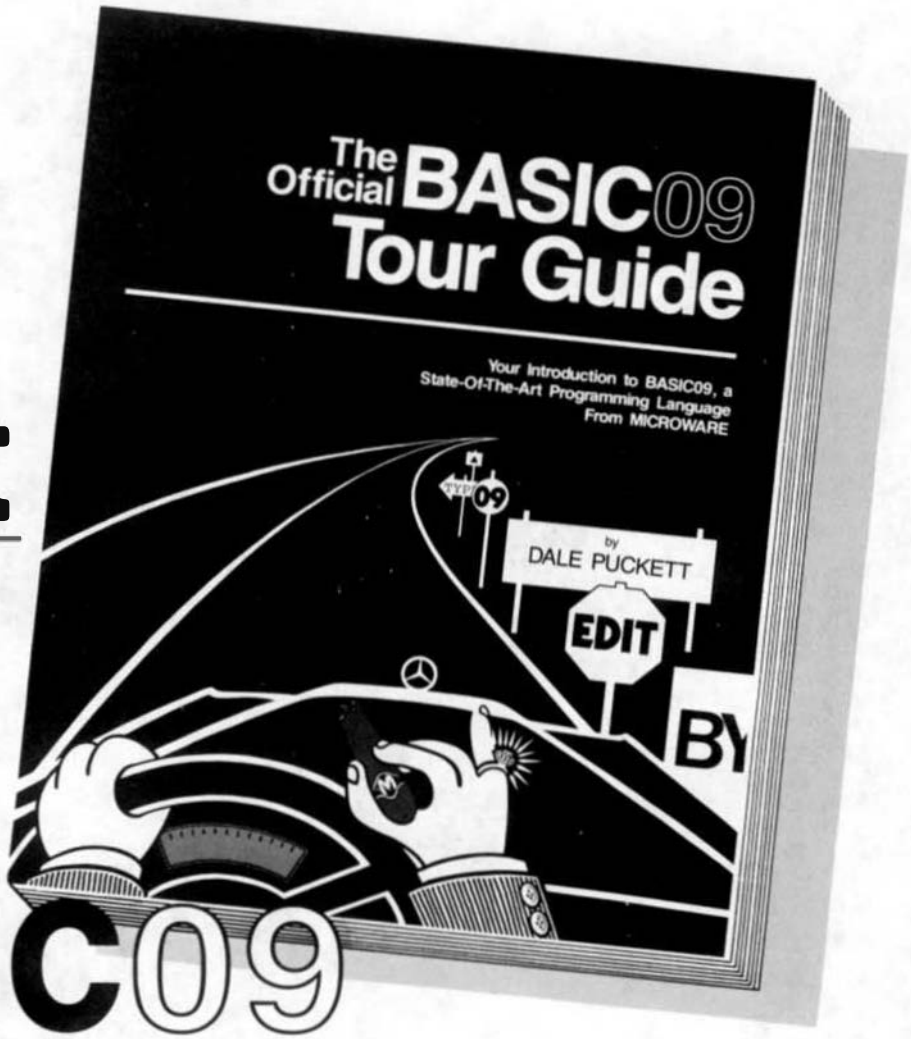
Please phone or write if you need further information.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc.
FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc.
GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

GIMIX inc.

1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609 • (312) 927-5510 • TWX 910-221-4055

Get the most out of BASIC09



The **OFFICIAL BASIC09 TOUR GUIDE** is skillfully written in a friendly and easy-to-read style. Just perfect for those new to computers and to BASIC09. It's also a *valuable reference book* for programmers, engineers, students and hobbyists, providing an in-depth look at BASIC09 plus an overview of the OS-9 operating system. Comprehensive reference sections on BASIC09 and OS-9 commands are also included.

The book "maps" out your route through the Mercedes of Basics . . . BASIC09 and puts you in the driver's seat in no time. Fasten your seatbelt, sit back and enjoy the ride to perfecting your programming skills.

MICROWARE . . .

The **OFFICIAL BASIC09 TOUR GUIDE** comes from the people who wrote BASIC09. As the leader in 6809 system software, we at MICROWARE care about our users and want to help you get the most from our products.

It's Easy to Order.

Phone orders are accepted from MasterCard or VISA cardholders or for COD shipment. You can also order by mail using the coupon below. Quantity discounts are available to educational organizations and dealers. For further information contact Microware.

microware®

Specialists in system software for 68-family microprocessors since 1977.

OS-9 and BASIC09 are trademarks of Microware and Motorola.

Microware Systems Corporation
1866 N.W. 114th Street
Des Moines, Iowa 50322
Telephone 515/224-1929
Telex 910-520-2535

Please send _____ copies of the **Basic09 Tour Guide** book at **$18.95** each. Add $2.00 for UPS shipping in the U.S. or $5.00 for overseas air mail per book. Iowa residents add 4% sales tax.

Name _____

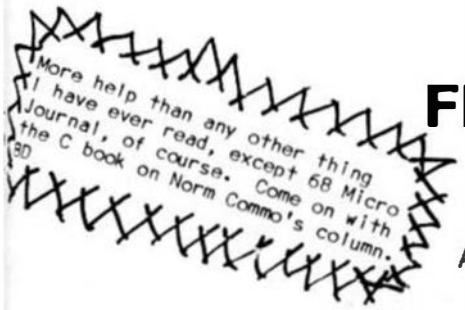
Address _____

City _____

State _____ Zip _____

☐ I have enclosed a check

☐ Charge to my bank card:
MasterCard ☐ VISA ☐
Card Number _____
Expiration _____



FLEX™ USER NOTES
THE 6800-6809 BOOK

By: Ronald W. Anderson
As published in 68 MICRO JOURNAL™

The publishers of 68 MICRO JOURNAL are proud to announce the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68 MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. Now all his columns are being published, in whole, as the most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

As a SPECIAL BONUS all the source listing in the book will be available on disk for the low price of: FLEX™ format only — 5" $12.95 — 8" $16.95 plus $2.50 shipping and handling, if ordered with the book. If ordered separately the price of the disks will be: 5" $17.95 — 8" $19.95 plus $2.50 shipping and handling.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC
	(And many more)

**Over 30 TEXT files included in ASM (assembler) — PASCAL — PIC (position independent code) TSC BASIC-C, etc.

NOTE: .C1,.C2, etc. = Chapter 1, Chapter 2, etc.

This will be a limited run and we cannot guarantee that supplies will last long. Order now for early delivery.

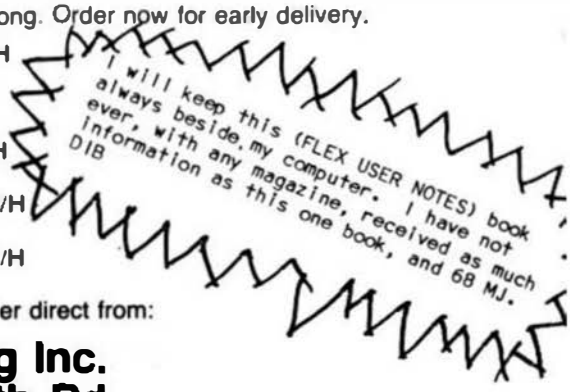
Foreign Orders Add $4.50 S/H

Softcover — Large Format

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H



See your local S50 dealer/bookstore or order direct from:

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4601

TELEX 558 414 PVT BTH

™FLEX is a trademark of Technical Systems Consultants

$4,325
FOR A WORLD-CLASS SS-50 COMPUTER

Smoke Signal's VAR/68™ gives you:

- Fabled Chieftain performance that led the pack in tough Benchmark surveys
- Integrated, easy-to-use software that covers your *complete* business needs
- Proven reliability backed by our exclusive Endurance-Certification Program
- Extremely good looks and unsurpassed operator comfort

$4,325: A PRICE CALCULATED TO GET YOU HOOKED ON THIS BLOCKBUSTER SS-50

That price buys you a VAR/68 computer with multi-user, multi-tasking capabilities, and an ergonomically designed terminal. You get 128K RAM—expandable to 1mb. Eight serial ports, up to 16 if desired. Two parallel ports—and more are available. Plus a long list of other impressive capabilities.

Smoke Signal's experience allows us to offer OS-9 and other UNIX-like, and multi-user operating systems.

The styling is completely new—fashioned for the utmost in operator comfort. And it's remarkably compact. VAR/68 is a combination of great performance and good looks demanded by the office of today.

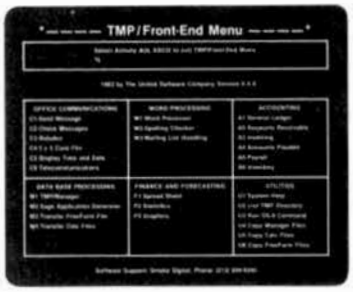
VAR/68 IS TOUGH, BUT SMOKE SIGNAL GIVES YOU *EXTRA* PROTECTION

(1) Your new computer is Endurance-Certified before delivery. That's an exclusive quality-assurance process that guarantees perfect operations from day one. (2) Our Advance-Replacement program is yours for a low fixed charge. (3) You get instant diagnostic service by telephone. It's free. (4) Normal repairs are handled with super speed. (5) Software *and* hardware support are part of doing business with Smoke Signal.

TOTAL INTEGRATED SOFTWARE GIVES YOUR BUSINESS *SOLUTIONS* INSTEAD OF PROBLEMS

Powerful business application programs are ingeniously interlinked to give even untrained operators a quick, smooth upper hand. The VAR/68 is a joy for first-time users, and an unprecedented productivity tool for anyone who wants new dimensions of control over critical business matters.

This screen tells part of the story:



GET A BIG DISCOUNT ON YOUR INITIAL ORDER

Most re-sellers can save up to 42 percent—even on small orders. Smoke Signal's price schedule is a powerful profit-maker for dealers of almost every description.

CALL SMOKE SIGNAL OR WRITE FOR MORE INFORMATION ON THE VAR/68 COMPUTER FAMILY

VAR/68™
BY SMOKE SIGNAL
The computer designed for your specialized application software

SMOKE SIGNAL
Products and Support for VARs

31336 Via Colinas • Westlake Village, CA 91362-3984 • (818) 889-9340

THE 68000 FROM SMOKE SIGNAL!

ADD 68000 AND UNIX™ * TO YOUR EXISTING SS-50 COMPUTER AT PRICES 50% TO 75% OFF LIST

THANK YOU

Seven years ago, Smoke Signal was founded to sell state-of-the-art computer products, by mail, to individual professional programmers and hardware engineers. At that time, most big companies did not believe in the power or future of microcomputers for serious computing applications. Only after you, the individual computer user, proved the viability of the microcomputer was Smoke Signal able to sell systems for business uses. However, as we progressed to become the leader in SS-50 systems, we had to add the sales and technical support services demanded by these business customers — and our prices for complete systems reflected these added costs.

With the introduction of our 68000 products, we wanted to find a way to say thanks to you, our original customers, the individual computer users, and still offer complete sales and technical support to our business customers for complete systems. We think this offer accomplishes both of these goals. We are offering you a choice of upgrade kits that will bring any SS-50 computer up to the electrical equivalent of our complete 68000 computer systems at prices far below complete system prices. In fact, the prices offered are 50% or more off our normally low prices for the components contained in the upgrade kits.

This special offer is limited to one upgrade kit per customer and is our way of saying thanks to those of you who had confidence in us from the beginning.

THE UPGRADES

The following upgrade kits were designed so that eny SS-50 system can be upgraded to 68000/UNIX.

SWTP UPGRADE..... $2,800.00
Contains: LMB-1A SS-50C Motherboard, DCB-4A floppy controller, PSA-1 Winchester/Tape DMA interface, SCB-68K 68000 CPU, SER-2 dual serial board, 5Mb Winchester and controller, power supply, all cables, and REGULUS.

GIMIX UPGRADE..... $2,500.00
Contains: Same as SWTP Upgrade except allows you to use your GIMIX motherboard, serial board and Winchester power supply.

Users of standard SMOKE SIGNAL systems may choose one of the following upgrade kits:

For SSB floppy based systems:

SS-FD UPGRADE $2,100.00
Contains: SCB-68K 68008 CPU, PSA-1 Winchester/Tape DMA interface, 5Mb Winchester and controller, power supply, all cables, and REGULUS.

For SSB Winchester based systems:

SS-HD UPGRADE..... $500.00
Contains: SCB-68K 68008 CPU and REGULUS.

COMPLETE SYSTEMS

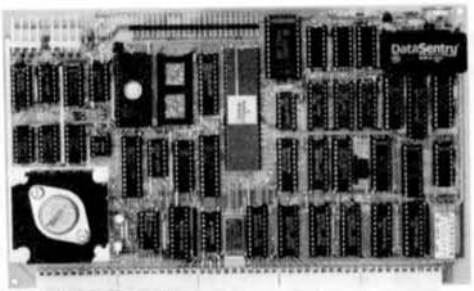
SMOKE SIGNAL is also making available complete VAR/68K™ systems at dramatic discounts. This offer is only available through **SMOKE SIGNAL** dealers. Contact **SMOKE SIGNAL** directly for information about how to order a complete VAR/68K system.

RULES OF THE OFFER

1) Limit, one upgrade system per customer.
2) Prices valid through December 31, 1984.
3) Orders must be accompanied by full payment in the form of individual check or credit card authorization.
4) Support will only be provided for systems containing the following **SMOKE SIGNAL** boards: SCB-68K, DCB-4A, PSA-1, and a motherboard such as the LMB-1A with extended addressing and main terminal I/O at FF7E8.
5) While we feel that most static RAM boards will work with these upgrades, we only guaranty compatibility with systems containing **SMOKE SIGNAL** static or dynamic RAM.

VAR/68K is a trademark of Smoke Signal.
REGULUS is a registered trademark of Alcyon Corp.; UNIX is a registered trademark of Bell Laboratories; OS9 and OS9/68K are trademarks of Microware; MACSBUG is a trademark of Motorola Inc.

***Regulus the OS offered is UNIX Compatible**



PRODUCTS

The heart of all these upgrade kits is SMOKE SIGNAL'S new SCB-68K 8 MHz 68008 CPU Board. This standard (5½" x 9") board will replace a SCB-69 CPU Board in any **SMOKE SIGNAL** computer with current revision boards. This board contains a real-time clock with battery back-up, 2 EPROM slots for up to 64K bytes of storage, a MACSBUG™ type monitor along with an auto boot loader and a mnemonic disassembler, plus many more features.

All upgrades also come standard with REGULUS™, a UNIX like operating system which is totally compatible with UNIX. REGULUS supports real-time tasks, shared memory, record locking and contains a shell similar to the Berkeley C shell. Along with the operating system, you get C, an editor, assembler, linking loader, interactive debugger and a word processor.

SMOKE SIGNAL is also including in many of the kits the DCB-4A double density floppy controller which can handle up to four 5" and four 8" floppies and contains 1K of buffer RAM for fast disk transfers; the PSA-1 Winchester/Tape DMA interface board which has taps for SASI and Priam disk interfaces as well as a tap for 90 ips tape streamers which are supported under both REGULUS and OS9™; either a M-256-X or M-512-X dynamic RAM board with over two years of field proven reliability; and the LMB-1A heavy duty motherboard with gold plated connectors, extended addressing and on-board baud rate generator with ten selectable baud rates.

SOFTWARE

Software and Software Support is available only from Smoke Signal dealers. Spread Sheet, Word-Processing, Relational Database, C, Basic and Cobol are all available now. Additional system's software is becoming available every day because of the UNIX compatibility.

SMOKE SIGNAL dealers are also offering Microware's OS9/68K™ to purchasers of these upgrade kits. SMOKE SIGNAL will offer other Microware 68000 products as they become available.

SUPPORT

Even at these "lower than PC" prices, we're not going to leave you with "PC" type support. We've arranged with one of our very technically qualified dealers to provide you with add-on software and technical support. In addition to answering your questions on how to convert your system to the 68000, he has a group of his customers who are themselves computer experts who are joining in a network that will help with even the most technical questions. We hope you will contribute your ideas to the network so that we all can benefit from new and fresh thinking. Complete details of the support available are included with the upgrade systems.

ORDER FORM

Fill in your name, address and phone number below. Your order will be shipped UPS so please do not use P.O. Box. Check items being ordered on form. Add prices for all items selected. CA residents must add 6% for sales tax. Total the amount for your order and check payment method below.

Name _____
Address _____
City, State, Zip _____
Phone _____

Payment: ☐ Enclosed Check
☐ VISA
Mastercard
Card # _____
Exp. Date _____
Signature _____

☐ SS-FD UPGRADE	$2100	_____
☐ SS-HD UPGRADE	500	_____
☐ SWTP UPGRADE	2800	_____
☐ GIMIX UPGRADE	2500	_____
☐ M-256X RAM	648	_____
☐ M-512-X RAM	948	_____
☐ SER-2 I/O	65	_____
☐ 20Mb HARD DISK (instead of 5Mb)	800	_____
Sub Total		_____
CA residents add 6%		_____
Total		_____

SEND COMPLETED ORDER FORM TO:
SMOKE SIGNAL
31336 Via Colinas, Westlake Village, CA 91362-3864
(818) 889-8340

Flex User Notes

Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

Letter

I have in front of me a letter that is dated May 17, 1984. I had gotten material rea y for a reply some time ago, but I wanted to have plenty of time to formulate my reply. I'll first present the letter verbatim.

"Dear Ron,

During the last months, I have been reading the different opinions on assembler versus HLL in your column. Depending on temperament some of the views advocated could either make you laugh or cry, but I think one, maybe the most important aspect in writing serious software, has been totally neglected. I am thinking of reliability /maintainability /serviceability.

What good is a piece of software that runs twice as fast as some other software if you have to scrap it after some time because it is impossible to repair? I for my part would hate to be the purchaser of some system running on software created from the 350 c mand monster mentioned in the May issue of '68 Micro. The program could probably only be serviced by its writer, but I bet even he would be in trouble if he had to come back to it in a couple of years.

What I am aiming at is that software and software based systems also should be looked at economy-wise. I suppose a big percentage of '68 Micro's readers make their living from software /systems, and if they don't take programmer efficiency and the reliability, testability, serviceability and maintainability of their software into account, they will soon have to find another way to earn their money, leaving customers with software that is not worth their media.

Incidentally all the points I have made so far are best taken care of using a good efficient portable language like 'c', and I really do not understand why you discourage porting software to new environments. Do you prefer to reinvent the wheel every time you try something new?

Your lack of knowledge of what an emulator can do is horrifying. Without an emulator you can in no way test your system under realistic conditions. You may be able to test the integrity of your software per se, but to say that program testing alone is sufficient is like testing the engine of a car later to discover that the car does not run because someone forgot the wheels.

To say that an emulator is just a tool for single-stepping your target machine is nonsense. It is a tool to observe the system while it is running full speed, at the same time keeping trace of all internal and some external action so that you, in case of trouble can find an explanation. Using just software debugging you have no way to cope with peripherals, external activity, access times, glitches and so on. Frankly speaking, if I were the manager of a company with engineers /programmers displaying points of view of the kind you have on the use of emulators, I should reconsider your employment.

I hope my reflections may give rise to further comments. They have intentionally been made somewhat provocative, but I do not think we should be confident just writing some apparently good software without thinking of the purpose of the same software. If it makes you happy to write a bit of software that runs faster, smaller, whatever, not giving any thoughts to how you write it, you should keep it to yourself and not try to market it.

Yours sincerely,

Niels Desten
Brostykkevej 189
DK-2650 Hvidovre
Denmark.

Response

First of all, thank you Niels for your letter. I am a bit put off by the fact that the way your letter reads, you seem to be attributing ALL of the points of view put forth in the discussion to me personally!

Now with regard to your first point, the subject of Reliability, Maintainability, et. al. I think you are correct in that this aspect of the subject was not given enough emphasis. The omission was not intentional. I know that is one of the major reasons for using a higher level language. The subject did come up in various comments throughout the series. For example in the November '83 column I said "Another programmer can read and understand our high level code much more quickly than he can understand our Assembler co e..." That does say something about maintainability, doesn't it? In the January '84 column, I said in reply to Dan Farnsworth's remarks:

"Another point that you made is that you had written a number of subroutines that you use in writing programs in assembler. In effect, you have written your own compiler in "secret code" that no one else can understand AS EASILY AS YOU. Using a standard language results in a program that may be understood by someone else quite easily... As programs become larger and larger, and reach the point where more than one person has to work on them, or when they must be maintained for customers over a long period of time, so that there is no guarantee that the original author will still be an employee of the company, (or for that matter, might be ill or even die), the use of a higher level language makes much more sense."

With regard to your comment about the 350 command monster, I fully agree that my 50 structured commands in a higher level language are superior by virtue of being easier to understand and remember. Now with regard to efficiency and the economic factors of pro ucing a piece of software, I think Art Weller in his letter to me that appeared in the March '84 column addressed that topic rather adequately.

"You didn't have to specifically say that the discussion has to do with productivity -- it's obvious. That will get a manager's attention every time and should, in fact, concern even a free lance programmer (isn't he his own management?"). To make it a little easier to get to the point, let's assume a situation in which the programming, computer operation, and 'user' are all 'in-house' (as it was in my case). Variations of this, as for example, production of software for sale (user=customer) don't appreciably change the problem.

"I can't think of any activity that is more 'labor intensive' than computer programming; can you? It consists of just about 100% labor and you can't even hold the product in your hand to examine it. Worse yet, it's usually very expensive labor (relative to the rest of the work force) and these costs must be recovered or amortized somewhere in the organization. That's just gobble-de-gook way of saying that programming effort must pay for itself by reducing costs elsewhere in the company. More efficient use of the computer, a reduction of workloa for a computer user/operator, or more effective use of some other resource."

I think that covers the "economic" consi erations to some extent at least. Your next paragraph indicates that you don't understand why I discourage porting software to new environments.... If the result of my labors were such sorts of applications software as screen editors, word processors, spread sheets, spelling checkers, and the like, I would certainly consider no other path than to use "C" or Pascal so they could be "ported" to ot`er systems. That decision would be strictly one of economics again! No one in his right mind (and trying to make a living by writing software) would spend a year or two writing the "perfect" editor for a market of 10,000 FLEX users, for example, when the writing of the same sofware in a portable language would result in a market of hundreds of thousands of computer users!

I think you had taken my remarks regarding portability out of context. What I said was that in my day to day work, I program stand-alone systems for an instrumentation application, part of a machine. I think the abstract example I used was a coffee vending machine. If I were writing software for the control of a coffee vending machine, I would want to pick the most economical route that provided easy maintainability of the code with respect to changing requirements in the machine. If such a goal could be met with a non-

standard language, I wouldn't hesitate to use it (as I do for my work) since it is not a disadvantage that my code can't be ported to another processor, but an advantage that my competitor can't somehow get a copy of the source code and run it on his "other processor" system. The point was and is, that for some situations portability is an asset. For others, it is not.

My point of view on an emulator has not changed since reading your letter. I still can't see where and how one would do anything for us that we can't do without it. I know, I oversimplified my discussion of the emulator when the subject came up. I realize that I can see where the program spends most of its time. That is usually obvious in looking at the program listing. If my program runs adequately fast, I really don't care where it spends most of its time. I know I can set breakpoints and examine past steps. That is useful only in debugging software. Timing problems? These can be avoided at least in the applications in which we are involved, by reading the supplier's data sheet carefully for (for example) an A/D converter. We don't hang an A/D on the bus, but communicate with it through a PIA. In critical areas of timing it is easy to count machine states to determine the length of the "convert command" pulse, and the delay between multiplexer address output and the start convert pulse. Besides, it is quite easy to look at this timing with a simple oscilloscope.

Perhaps, Niels, you are developing computer hardware and pushing the state of the art with regard to speed. If that is the case, an emulator would be extremely useful. Our hardware is all developed and debugged. We've successfully communicated with peripherals to read limit switches and controls, provide outputs to control machine functions, read position via linear analog transducers (potentiometers and Linear Variable Differential Transformers), measure motions (vibration) down to the ten micro inch amplitude level, read incremental encoders, interface to multi-horsepower servo motors, display subsystems, and probably a dozen others peripherals.

We have a system whereby we can "download" our software directly into RAM in the target system and run it. If we have a problem, we can lengthen or shorten time delays with a two minute edit, recompile, download procedure and try again. We've never had a "puzzling" timing problem except in the application of a particular A/D converter that would seemingly foul up right around half scale (which in our application corresponded to zero!). The next data sheet from the manufacturer of the A/D recognized the problem and provided a simple hardware solution. The problem was one of synchronizing the convert pulse to the local clock used by the successive approximation A/D converter!

Under the circumstances, I would respond in kind, Niels, that if you came to my company looking for a job and told me that you had to have an emulator or digital logic analyzer to perform your function, I would probably think twice before offering you a job.

In retrospect, I think most of the disagreement that has been aired in this column over the great Assembler - Compiler debate has come from the fact that we each are into a different area of computer applications. Let me once again give some hypothetical examples to prove the point.

1. I am involved in a project in which a Fast Fourier Transform must be performed in real time, on the data from the previous machine cycle. Maximum machine cycle rate is one cycle in two seconds. The most important goal that must be met in this case, is that the analysis calculation MUST take place in less than two seconds. If this goal can't be met, maintainability, economy, size of program, etc. are all irrelevant! The primary consideration in this case is SPEED!

2. I am designing a controller for a microwave oven. The production quantity is to be 500,000 over the life of the product. If I can save one dollar on parts, the company saves $500,000. Obviously, I will have to code in Assembler if I can save one ROM and/or simplify the address decoding because the program is shorter. The primary consideration in this case is COST.

3. I'm writing software for NASA for a Mars probe. Obviously, the constraints are several, that of efficiency in terms of speed and memory usage, RELIABILITY and MAINTAINABILITY as requirements change are probably the highest on the list. Who cares abo t portability in this sort of application?

4. I am writing the worlds greatest screen editor. Not only are reliability and portability important, but the editor has to work with anything from a "dumb terminal" through the latest "whiz bang" that does everything but the computation for you. Adaptability is of ultimate importance here. The primary concern here is PORTABILITY and FLEXIBILITY.

You see, writers of applications software for different purposes, quite naturally have goals that emphasize different aspects of the discussion. Just because you write general applications software such as editors and word processors, and maintainability and portability are highest on your priority list, doesn't mean that these aspects are at the top of the list for everyone's computer applications.

Rather than prattle on endlessly, I'll stop there. I think you are all bright enough to get the point. If anyone else has some response to Niels' letter (or to my "defense") please feel free to write and I will add your comments to those already presented.

Language Philosophies

I have been thinking about the differences in the philosophies of the different languages I have been using lately. The thing that prompted this was my working with Whimsical in preparing the review of it. There seem to be two distinct philosophies of language design. Permit me some license here to exaggerate considerably to make the distinction very clear.

The first philosophy is that the programmer is an absolute idiot who can't keep track of anything for himself. The compiler should therefore catch as many errors as is possible. If the programmer wants to do anything unusual he will have to tell the compiler that he knows what he is doing. Pascal and Whimsical fall into this category. For example in Pascal, suppose you want to output a carriage return. WRITE (CHR(13)); will do that without adding the linefeed that you would get with a simple WRITELN; statement. Note that the function CHR in Pascal does not generate any code. It must be used to tell the compiler that you know that you are telling it to output a decimal number as a character. In other words 13 is of type INTEGER and you want to write a CHAR to the terminal, so you must tell the computer to convert the INTEGER to a CHAR.

The other philosophy is that the programmer is good for something, and that he has some capability to think about what he is doing. If he writes a statement that outputs a decimal integer as a character, that is what he wanted to do, so it is OK. That is the philosophy of "C" and PL/9. putchar(13); is a perfectly acceptable statement in "C". Since putchar requires a character for the parameter to be passed to it, the conversion is done automatically from a 16 bit integer to an 8 bit character. PL/9 doesn't have a data type CHAR at all. It has a type BYTE, and BYTE is used for handling characters.

Another example of cheating might be a way to "clear" variables in a "C" or PL/9 program. Suppose, for example, that you have declared an array of integer of dimension 10 (index range 0 to 9 in PL/9 and "C"), and that you have declared three simple integer variables immediately following the array. If you are sure that variables are allotted space in the order in which they are declared, you can do something like:

```
INDEX = 0;
WHILE INDEX <13
BEGIN
   INTARAY(INDEX)=0;
   INDEX = INDEX+1;
END;
```

The result of that is not only to clear the array but to clear the three integer variables that follow the array. Why resort to such a trick? The code generated is independent of the loop termination value, and you save three assignment statements to set the three integer variables to zero. The program runs faster and uses less memory. If you are careful to include the comment, "CLEAR ARRAY AND THREE FOLLOWING INTEGER VARIABLES", is the code any less clear? Try that in Pascal and you will get an "ARRAY INDEX OUT OF RANGE" error immediately.

I can think of another useful trick to squeeze some execution time out of a program. Suppose you have an array of BYTES of dimension 100. In PL/9, at least, you can declare an array of INTEGER of dimension 50 at the same location as the first array. Now you can clear or

assign the value of zero to two bytes at a time. The clear loop only has to execute 50 times, and will almost certainly run considerably faster.

Yes, these are the tricks of the assembler programmer. One learns them by spending time programming in assembler and reading other programmers' code. As I said above in the response to the letter, and have said on many other occasions in this column, which is best depends primarily on the specific application. There must be a compromise usually between maintainability, reliability, execution speed, memory efficiency, difficulty of writing the program, and other factors. The specifics of the problem at hand will determine in which direction the compromise must be biased best to solve the problem. Some problems bias the compromise totally, as in a case where the program must execute in absolute minimum time, or in the case where reliability is of such overwhelming importance that none of the other factors matter at all.

To sum it all up again, there is no best language for all purposes. Assembler is not always the best way to go. Strictly typed languages are not always best, and loosely typed languages are not always the best. The best and most successful programmers realize that they have a whole spectrum of tools in the form of programming languages, and they will choose the best one for the job at hand by carefully evaluating the requirements and choosing a language that most nearly meets them.

I suppose my own position is fairly obvious from what I've said so far. Yes, I prefer the languages that will let me "cheat" and allow me more concise program statements. I think, however, that before you use one of these less rigid languages, you ought to become an expert in Pascal. You will soon learn to appreciate the value of writing code in short segments or modules, and you will see the huge advantage of keeping the modules as independent from each other as possible by minimizing their interaction. Though "C" and PL/9 do allow you to "cheat", they are modern languages that have facilities for local variables and parameter passing to procedures or functions. BASIC has neither the philosophy of forcing structured code nor the facilities to allow it. It doesn't allow meaningful variable names, it doesn't allow named subroutines or functions, it has only global variables, and it lacks all of the loop control structures such as DO WHILE and REPEAT UNTIL, so that you must use a GOTO to control such loops. Still, no one would argue that BASIC is not a useful language. You can simulate the above loop control structures with a comparison and a GOTO statement, and indicate what you are doing by means of a REM statement.

This discussion falls right in line with the letter above. Niels (I think) would opt for Pascal or a similar language that leaves nothing to chance or the whim of the programmer. Such rigid languages ARE essentially "self documenting" so that comments are less essential. The nature of the language requires documenting in the form of the program statements and variable declarations themselves.

I realize that few computer hobbyists (and not too many professional programmers) are as fortunate as I with regard to being able to try out the various language implementations firsthand. Perhaps these thoughts will at least narrow down the choices for you and your range of applications. By all means, don't take anyone else's advice too seriously, who thinks he has the one and only answer to any and ALL programming needs.

Incidentally, I have not mentioned FORTH here for some time. It is most definitely not the language for me and my applications, but you might think it is perfect for yours. By all means investigate it and try it out.

- - -

SUPPORT YOUR ADVERTISERS

OS9 USER NOTES

By: Peter Dibble
517 Goler House
Rochester, NY 14620

Remote Procedure Call Almost

I devoted a great deal of time to interprocess-communication about two years ago. This is a pet project of mine. It's relatively hard, useful, and underused. The methods I have presented in the past are flexible but need to be adapted for each problem. In this column I'll show you a less powerful, lower performance method that can be used without any particular thought. It can be smoothly built into a program in Basic09 or C (or assembler, of course). The best way to use "rpc" would be to build it into a programming language.

What's a Remote Procedure Call

Properly, a r note procedure call is a sort of subroutine call that can pass between machines. If you have two OS-9 systems attached to one another and my tool extended as far as its name implies, you could run a program on one machine and call subroutines on the other. If you look carefully at rpc you'll see that it was designed for this but needs some lower-level support before it can reach between machines. "Rpc" as is stands is more of a cross-memory procedure call, but I decided to name it ambitiously.

Calling a subroutine through rpc is MUCH slower than a normal procedure call. I haven't measured, but the overhead involved in rpc is probably at least a thousand times as much as a regular procedure call -- maybe hundreds of thousands. It would be a impressively bad idea to design a program that used rpc to call a small subroutine that gets a lot of use. A more appropriate use would be to connect passes of a compiler. If you figure on using rpc about every ten seconds you won't notice any performance problem.

Comparing rpc to a regular procedure call isn't strictly fair. If you can use a procedure call you should. Rpc is a high-powered tool for those times when you can't fit your entire program in one address space, or a slick way of connecting a series of programs without temporary files or explicit pipes. Compared with running a string of programs connected by temporary files (like the C compiler) rpc is very fast.

Some Warnings

Since rpc isn't hooked into a compiler it doesn't have protection like "strong typing." You have to take extra care when you use it. Parameters to rpc, rpe, and rpx MUST match. If you use a variable with the wrong length the program will come to a halt. You can also get into trouble by simply forgetting to call rpe and rpx in the subroutine.

Rpc requires two standard parameters: the name of the module to call and the parameter string to use with the F$Fork. The parameter string for F$Fork must be used if the subroutine is a Basic09 program; see the Basic09 example. The parameter string also offers some possibilities for pass-by-value that I won't get into. After the

parameter string you can put as many arguments as you like. All these arguments will be copied for the called procedure, then copied back with any modifications. They must all be pass-by reference.

The subroutine should call rpe to get the parameters. You don't have to call rpe (remote procedure enter) first thing, but there's no point in waiting. The parameters aren't available until you do (except enything you passed in the F$Fork parameter string.). When you are ready to return the parameters to the caller use rpx (remote procedure exit).

It should be possible to use these modules (rpc, rpe, rpx) from C. They preserve Y and U, and don't require anything C can't provide. To make calling them from C easy, recode them into RMA. The call would look something like

```
rpc(ArgCount, ModName, ModNameLen, Param, ParamLen,
    Arg1, Arg1Len, Arg2, Arg2Len, ...)
rpe(ArgCount, Arg1, Arg1Len, Arg2, Arg2Len, ...)
rpx(ArgCount, Arg1, Arg1Len, Arg2, Arg2Len, ...)
```

ArgCount is the number of Basic09 arguments you are faking; e.g., for an rpc call with two passed parameters it would be 4. All the "Args," the ModName, and Param should be passed as addresses.

To run the demonstration program, pack the Basic09 procedures and load them with the OS-9 load command, load the file containing the rpc modules, and type

```
OS9:basic09 caller
```

\- - -

```
                        ttl   a part of Remote Procedure Call
                        nam   rpc
*--------------------------------------------------------------*
*  rpc is a subroutine for basic09 and C                       *
*  It is based directly on StrtTask                            *
*  Start a named module as a subtask.                          *
*    Let the new task run asynchronously.                      *
*    Open pipes to the modules standard in and standard        *
*    out paths.                                                *
*    Return the new tasks process number, the path             *
*    numbers for the pipes, and the condition code             *
*    from the fork.                                            *
*  Calling sequence:                                           *
*  rpc (Name, parms, arg1, arg2, arg3,...)                     *
*  rpc(argcnt,name,namelen,Parms,ParmSize                      *
*      arg1,arg1 len.....)                                     *
*  the arguments must all be passed by reference.              *
*  Name is any length, but has a valid terminator              *
*   (high bit set on last byte, or delimiter after it)         *
*  Parms are passed to the new process as a parameter          *
*  area.                                                       *
*                                                              *
*  Each of the other fields is copied through standard         *
*  input to the called program, then read back from            *
*  standard output.                                            *
*  The called program must call rpe to get the                 *
*  parameters and rpx before terminating to send them          *
*  back.  Awful things happen if the variables in              *
*  rpc, rpe, and rpx don't match in number and size.           *
*--------------------------------------------------------------*
                        IFP1
                        EADC
******
*  Offsets to arguments
*
0000 D 0000                 org   0
*  Offsets from S for local storage
0000 D 0000 OStdIn      rmb   1
0001 D 0001 OStdOut     rmb   1
0002 D 0002 TaskNo      rmb   1
0003 D 0003 LocalSiz    equ   .
*
*  space for push of Y,U
*
0003 D 0003 RY          rmb   2         push Y
0005 D 0005 RU          rmb   2         push U
*
*  formal parameters
*
0007 D 0007 RetPC       rmb   2
0009 D 0009 ArgCount    rmb   2
000B D 000B ModuleN     rmb   2
000D D 000D ModulePC    rmb   2
000F D 000F Parms       rmb   2
0011 D 0011 ParmLen     rmb   2
0013 D 0013 ArgStart    equ   .
*
0021        Type        set   SBRTN+OBJCT
0081        Revs        set   REENT+1
0000        $stdin      equ   0
```

```
0001        StdOut      equ   1
0000 87CD0028           mod   TLen,rpc,Type,Revs,SEntry,0
0000 7270E3 rpc         fcs   /rpc/
0010 01                 fcb   1             version
0011 2F5D4950 Pipe      fcs   "/PIPE"
0016        SEntry
0016 3460               pshs  Y,U           save environment
0018 3270               leas  -LocalSiz,S make space for temp storage
                        *--------------------------------------*
                        *      Set up Pipes for Stdin and StdOut.*
                        *  The procedure is:                   *
                        *    Dup the stdin and stdout paths to save them.*
                        *    Close stdin and stdout.           *
                        *    Open /PIPE twice. One will be path 0 the next*
                        *      path 1.                         *
                        *    Fork the new process.             *
                        *--------------------------------------*
001A B600               lda   $Stdin
001C 103F82             OS9   I$Dup     Dup Stdin
001F 10250090           lbcs  BadExit
0023 A7E4               sta   OStdIn,S
0025 B600               lda   $Stdin
0027 103F8F             OS9   I$Close   Close Stdin
002A 10250091           lbcs  BadExit
002E 308DFFD8           leax  Pipe,PCR
0032 8603               lda   #UPDAT.
0034 103F84             OS9   I$Open    Open a pipe in stdin
0037 102530B8           lbcs  BadExit
003B B601               lda   $StdOut
003D 103F82             OS9   I$Dup     Dup StdOut
0040 10250078           lbcs  BadExit
0044 A761               sta   OStdOut,S
0046 B601               lda   $StdOut
0048 103F8F             OS9   I$Close   Close StdOut
004B 10253070           lbcs  BadExit
004F 308DFFB6           leax  Pipe,PCR
0053 8603               lda   #UPDAT.
0055 103F84             OS9   I$Open    Open a pipe in stdout
0058 10250063           lbcs  BadExit
                        *
005C AE88               ldx   ModuleN,S address of module name
005E 10AEE811           ldy   ParmLen,S length of parameters
0062 ECE811             ldd   ParmLen,S optional data area size (0 if)
0065 C300FF             addd  #$FF      round up
0068 1F89               tfr   A,B       save high-order byte in B
006A 8603               lda   #0        type of module to invoke (valid
006C EE6F               ldu   Parms,S   pointer to parameters
006E 103F03             OS9   F$Fork    start the new process
0071 1025004A           lbcs  BadExit
0075 A762               sta   TaskNo,S  save the childs task number
                        *--------------------
                        *  Transmit the arguments to the called procedure
                        *
0077 33E813             leau  ArgStart,S pointer to args
007A E66A               ldb   ArgCount+1,S
007C C002               subb  #2        subtract for modulen and Parms
007E B600               lda   $Stdin    path number for transfer
0080        SendLoop
0080 3730               pulu  X,Y
0082 103F8A             OS9   I$Write
0085 2538               bcs   BadExit
0087 5A                 decb
0088 26F6               bne   SendLoop
                        **********
                        *  Recover the arguments from the called procedure
                        *
008A 33E813             leau  ArgStart,S pointer to args
008D E66A               ldb   ArgCount+1,S
008F C002               subb  #2
0091 B601               lda   #StdOut
0093        GetLoop
0093 3730               pulu  X,Y
0095 103F89             OS9   I$Read
0098 2529               bcs   BadExit
009A 5A                 decb
009B 26F6               bne   GetLoop
                        *--------------------------------------*
                        *    Restore the original stdin and stdout files to*
                        *    paths 0 and 1.                    *
                        *--------------------------------------*
009D B600               lda   $Stdin    Close Stdin
009F 103F8F             OS9   I$Close
00A2 A6E4               lda   OStdIn,S  path number of duped stdin
00A4 103F82             OS9   I$Dup     dup it into path 0
00A7 A6E4               lda   OStdIn,S
00A9 103F8F             OS9   I$Close   close the dup of stdin
                        *
00AC B601               lda   $StdOut
00AE 103F8F             OS9   I$Close
00B1 A681               lda   OStdOut,S path number of duped stdout
00B3 103F82             OS9   I$Dup     dup it into path 1
00B6 A661               lda   OStdOut,S
00B8 103F8F             OS9   I$Close   and close it
                        *
00BB 103F04             OS9   F$Wait    wait for the process to comple
00BE 5F                 clrb            clear carry
00BF        BadExit
00BF 3263               leas  LocalSiz,S clear stack
00C1 35E0               puls  Y,U,PC    return
00C3 ACFE14             CMPD
00C6        TLen        equ   *
                        *
                        nam   rpe
                        ttl   Remote Procedure Enter
0000 87CD0079           mod   ELen,rpe,Type,Revs,EEntry,0
0000 7270E5 rpe         fcs   /rpe/
0010 01                 fcb   1             version
                        *
0000 D 0000             org   0
0000 D 0000 ERY         rmb   2
0002 D 0002 ERU         rmb   2
                        *
0004 D 0004 EPC         rmb   2
0006 D 0006 EArgCt      rmb   2
0008 D 0008 EArgStrt    equ   .
                        *
0011        EEntry
0011 3460               pshs  Y,U
0013 E867               ldb   EArgCt+1,S
0015 B600               lda   $Stdin
0017 3368               leau  EArgStrt,S
```

```
00187
00188  0019                 EmLoop
00189  0019 3730                    pulu  X,Y
00190  001B 103F89                  OS9   I$Read
00191  001E 2504                    bcs   EBad
00192  0020 5A                      decb
00193  0021 20F6                    bne   EmLoop
00194
00195  0023 3F                      cirb
00196  0024                 EBad
00197  0024 35C0                    puls  Y,U,PC
00198  0026 2E42A7                  EMOD
00199  0029                 ELen    equ   *
00200
00201                               nam   rpc
00202                               ttl   Remote Procedure Exit
00203  0000 87CD0079                mod   XLen,rpc,Type,Revs,XEntry,0
00204  0000 7270FB          rpc     fcs   /rpc/
00205  0010 01                      fcb   1             version
00206 D 0000                        org   0
00207 D 0000                 XRY    rmb   2             push Y
00208 D 0002                 XRU    rmb   2             push U
00209 D 0004                 XPC    rmb   2             return address
00210 D 0006                 XArgCnt rmb  2
00211 D 0008                 XArgStrt equ  *
00212  0011                 XEntry
00213  0011 3460                    pshs  Y,U
00214  0013 E667                    ldb   XArgCnt+1,S
00215  0015 8601                    lda   #StdOut
00216  0017 3368                    leau  XArgStrt,S
00217
00218  0019                 TLoop
00219  0019 3730                    pulu  X,Y
00220  001B 103F8A                  OS9   I$Write
00221  001E 2504                    bcs   XBad
00222  0020 5A                      decb
00223  0021 20F6                    bne   TLoop
00224
00225  0023 3F                      clrb
00226  0024                 XBad
00227  0024 35E0                    puls  Y,U,PC
00228  0026 FD17AB                  EMOD
00229  0029                 XLen    equ   *

00000 error(s)
00004 warning(s)
 6011B 00280 program bytes generated
 00023 00035 data bytes allocated
 32401 09217 bytes used for symbols
```

```
PROCEDURE caller
0000      DIM counter:INTEGER
0007      DIM message:STRING[20]
0013      DIM modulename:STRING[10]
001F      DIM parms:STRING[10]
002B      modulename="BasicO9 "
003A      parms="TestSub"+CHR$(13)
004C      message="abcdefghijklmnop"
0063      PRINT "Transmitted message is: "; message
0083      RUN rpc(modulename,parms,message)
0097      PRINT "Returned message is: "; message
00B4      END
PROCEDURE testsub
0000      DIM message:STRING[20]
000C      DIM errnum:INTEGER
0013      DIM i:INTEGER
001A      DIM remessage:STRING[20]
0026      RUN rpc(message)
0038      remessage=""
0037      FOR i=LEN(message) TO 1 STEP -1
0048        remessage=remessage+MID$(message,i,1)
0081      NEXT i
008C      RUN rpc(remessage)
0076      END
```

"C" User Notes

Edgar M. (Bud) Pass, Ph.D.
1454 Latta Lane
Conyers, GA 30207

INTRODUCTION

This month's column discusses the inconsistencies in the 6809 Full C memory allocation library functions. It also continues the discussion of string processing in the C language started in an earlier column, including a discussion of data compression and expansion.

MEMORY ALLOCATION INCONSISTENCIES

The more sophisticated string processing, list processing, and other data management techniques require the use of the C language memory allocation and deallocation library functions.

Unfortunately for the portability of C programs, there is a significant amount of inconsistency among the library functions provided by the various C compilers. The following table provides a summary of the memory allocation and deallocation library functions provided by several C compilers of interest to 6809 and 68000 users.

Windrush FLEX C	brk, sbrk
MPD UNIFLEX C	brk, cdata, sbrk
Microware OS/9 C	calloc, free, malloc
Introl C	alloc, free, malloc, sbrk
UNIX C	brk, calloc, free, malloc, realloc, sbrk

Following is a brief explanation of each library function just listed:

brk(address) requests that the program's addressable data space should include "address".

calloc(n,sz) allocates space for an array with "n" elements, each of size "sz", and initializes the space to 0x00.

cdata(address) requests contiguous allocation of the program's addressable data space to "address".

free(ptr) frees the space (pointed to by "ptr") previously allocated by "malloc" or "realloc".

malloc(sz) allocates a block of at least "sz" bytes.

realloc(ptr,sz) frees the space (pointed to by "ptr") previously allocated by "malloc" or "realloc" and allocates a block of at least "sz" bytes.

sbrk(sz) requests that the current program data area be increased by at least "sz" bytes, and initializes this space to 0x00.

The library functions just described return -1, 0, or NULL if they are unable to complete the requested task. They are thus inconsistent on their error return values.

The McCosh C compilers for the 6809 are inconsistent even among themselves. Only the OS/9 version has the "free" and "malloc" functions, which are the most useful memory deallocaton and allocation library functions.

The Introl C compilers for the 6809 are consistent among themselves. They all provide the "free" and "malloc" library functions.

Of course, the UNIX C compiler provides all the major library functions of the other compilers, and adds some of its own.

All of this inconsistency is amazing, as the UNIX C compiler and the K & R book were both available before any of the 6809 C compilers were developed. Certainly the promise of program portability is not fulfilled in this case.

Luckily, K & R provides examples of the coding of "free" and "malloc" functions, which the user may copy and adapt to individual C compilers missing the functions in their libraries. The Introl C library manager allows new pre-compiled functions to be installed into the library for automatic calling, but McCosh C does not provide a library manager. The unfortunate part is that this action is needed at all.

STRING PROCESSING IN C

When writing programs designed to process large amounts of text in limited space, the compression of text becomes an important and powerful method. Like most other techniques, there are many manners in which to perform data compression and expansion.

The technique described below is known as "radix translation". Text characters are restated into a smaller radix than their original representation. Thus, if 8-bit characters are restated into 5-bit characters, three compressed characters could be placed into the same space required by two uncompressed characters, with one bit to spare, which could be used as a flag. Radix bases are not limited to powers of two, but are often restricted to powers of two on binary computers, for purposes of efficiency and simplicity.

The primary limitation on the reduction of radix is the restriction of the number of characters which may be represented directly. Only 32 characters may be directly represented in a 5 bit radix, 64 in a 6 bit radix, 128 in a 7 bit radix, etc. Many more characters may be indirectly represented with the use of shift characters, as in Baudot 5-level code, but that level of complexity is not contemplated in the explanation below.

In terms of C string processing, compressed strings will, in general, contain arbitrary 8-bit values, including zero. Thus, C programs may not use any of the null-delimited string functions for moving the strings from one area to another, and must use length-delimited or some other context-sensitive logic to determine the end of a compressed string.

As a compromise between number of characters directly represented in a radix and the space required to represent the characters, radix base 40 is often used. This base represents 39 characters plus a terminating delimiter, or 40 characters with external length. Base 40 is used because 40 * 40 * 40 = 64000, which is expressible in 16 bits, a convenient unit for most modern binary computers.

This technique is far easier to implement than variable-radix techniques, such as Hamming code, does not require information about the relative frequencies of characters within a text file being compressed, and does not require a separate table for each file for the corresponding expansion.

The C program listed below performs compression/expansion in radix base 32 to 40. The comments describe its operation.

```
/* string compression/expansion (base 32 to 40) */

#include "stdio.h"
#include "ctype.h"

#define BASE 40 /* compression base 32 to 40 */

main()
{
    char c[255], ctab[256], *etab;

    /* assumes 16-bit integers */
    unsigned int com[174], i, j;

    /* load compression/expansion table
       with desired character codes */
    etab =
    "\0ABCDEFGHIJKLMNOPQRSTUVWXYZ .,!?\"#$%&'()*";

    /* load compression translation table
       such that untranslatable characters
       will be dropped and lower case
       letters will be mapped to upper case */
    for (i = 0; i < 256; ++i) ctab[i] = BASE;
    for (i = 0; i < BASE; ++i)
    {
        ctab[etab[i]] = i;
        if (isalpha(etab[i]))
            ctab[tolower(etab[i])] = i;
    }

    /* get strings from standard input */
    while (gets(c) != EOF)
    {
        /* terminate on null line */
        if (!*c) break;
        /* print lengths before and after
           compression, and print string */
        printf("%3d %3d %s\n", strlen(c) + 1,
            (i = compress(c, com, ctab)) * 2, c);
        /* print compressed string in hex */
        for (j = 0; j < i; ++j)
            printf("%04x ", com[j]);
        printf("\n");
        /* print expanded length and string */
        printf("%3d --- %s\n",
            expand(com, c, etab) + 1, c);
    }
    exit(0);
}

/* compress data from first parameter into
   second and return number of integers
   required to contain compressed data */
compress(unc, com, ctab)
char *unc, *ctab;
unsigned int *com;
{
    char c;
    unsigned int i, j, *fcom = com;

    do
    {
        for (i = j = 0; i < 3; ++i)
        {
            /* assumes 0 is translatable */
            if (!(c = *unc++)) --unc;
            /* skip untranslatable bytes */
            if (ctab[c] >= BASE)
                --i;
            else
                j = j * BASE + ctab[c];
        }
        *com++ = j;
    }
    while (c);
    return (com - fcom);
}

/* expand compressed data from first parameter
   to second and return expanded length */
expand(com, exp, etab)
```

```
unsigned int *com;
char *exp, *etab;
{
    char c, *fexp = exp;
    unsigned int i, j, b = BASE * BASE;

    do
        for (i = 0, j = *com++; i < 3; ++i)
            if (*exp++ = c = etab[j / b])
                j = (j % b) * BASE;
            else
                --exp;
    while (c);
    return (exp - fexp);
}
```

When the following file is processed by this program:

```
AAAAA
a*a*a*a*a
abcdefghijklmnopqrstuvwxyz
```

The following output is produced:

```
  6   4 AAAAA
0669 0668
  6 --- AAAAA
 10   4 a*a*a*a*a
0669 0668
  6 --- AAAAA
 27  18 abcdefghijklmnopqrstuvwxyz
0693 19ce 2d09 4044 5374 66ba 79f5 8d30 a050
 27 --- ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Note that 27 characters (26 letters plus null) is compressed into 18 bytes, then expanded to something closely resembling the original 27 characters, in the last example. This illustrates the two bytes for three data compression using radix base 40. Also note that the asterisks in the second example string are ignored in the compression, as they are not included in the compression/expansion table.

By changing the definition of BASE in the program above, the compression process could be made to use radix base 32. This could be of use in cases involving assembler-language expansion routines, since it is far easier to extract 5-bit fields than to divide by 40 on almost all microprocessors.

For example, the standard Motorola 6809 instruction set could easily be restated in base 32, since it requires only the letters, the digits 2 and 3, and terminator characters. The Motorola 6800 instruction set requires only the letters and a terminator character; futhermore, the fixed-length three-character mnemonics could be compressed into two bytes each, with no delimiter required. In either case, the compression/expansion table could be loaded as follows:

```
etab = "ABCDEFGHIJKLMNOPQRSTUVWXYZ23 \0\0\0";
```

and the radix base definition would be changed to BASE 32.

When the following file is processed by this version of the program:

```
abcdefghijklmnopqrstuvwxyz23
abx adc add and asl asr bit clr cmp
```

The following output is produced:

```
 30  20 abcdefghijklmnopqrstuvwxyz23
0022 0c85 18e8 254b 31ae 3e11 4a74 56d7 633a 6f9f
 30 --- ABCDEFGHIJKLMNOPQRSTUVWXYZ23
 36  24 abx adc add and asl asr bit clr cmp
0037 7003 0bb0 0c7c 01a3 7012 2f90 4a3c 0513 704b 4762 31ff
 36 --- ABX ADC ADD AND ASL ASR BIT CLR CMP
```

which illustrates the compression/expansion process, in radix base 32, using the revised compression/expansion table.

To demonstrate how easily 5-bit base radix compressed strings may be expanded, consider the first compressed set "0022" above. Rewriting "0022" into binary and regrouping into 5-bit groups, it becomes the following:

```
0 00000 00001 00010
```

Using each of the 5-bit groups as an index into the compression table, the expansion string becomes "ABC", which is correct. Alternately, the use of the expansion table could be avoided by noting that the first 26 entries are upper-case letters, represented in ASCII by the values 0x41 thru 0x5a, and the other compressed values could be checked for individually.

A routine could easily be written which searches a compressed radix base 32 or 40 string by expanding only one 16-bit set at a time, thus minimizing the space required, as only one 3-character work area would be needed at any point in time during the search. Alternately, the string being searched for could be compressed, and the routine could search for the combinations in radix base 32 or 40 directly.

C PROBLEM

Last month's problem was to write a program which will replace all sequences of multiple spaces and tabs with single spaces. There is no unique manner in which to solve this problem, but the following program will do so:

```
#include "stdio.h"
main()
{
    int c;
    c = getchar();
    while (c != EOF)
    {
        putchar(c);
        if ((c == ' ') || (c == '\t'))
            do
                c = getchar();
            while ((c == ' ') || (c == '\t'));
        else
            c = getchar();
    }
    exit(0);
}
```

Ask yourself why the pr ram would not work on some C compilers if the "int c;" declaration were changed to "char c;"; after all, the "getcher()" function returns the next character on the standard input.

Without using any of the string processing functions described earlier, write a C version of the BASIC "instr" function. It is defined as follows:

> instr(n,s,t) returns the character number of the first occurrence of string "s" in string "t" (starting with character number "n" in "t") or zero, if "s" does not occur in "t".

EXAMPLE C FUNCTION

Following is this month's example C function; it trims whitespace from either or both ends of a string. It is from Eric Martz of Amherst, MA.

```
/*
   Trims whitespace (spaces, tabs, newlines)
   off side ('r' right, 'l' left, 'b' both) by
   moving pointers "first" and "last", which
   point to the first character and ending
   null of a string, respectively.  It may be
   called as follows:
       trimwhite('b', &first, &last)
   in which "first" and "last" are pointers.
*/
trimwhite(side, first, last)
char side, **first, **last;
{
    if ((side == 'r') || (side == 'b'))
    {
        /* point last to end of string if it
            is passed with null value. */
        if (*last == NULL)
            *last = *first + strlen(*first);
        /* scan string backward for whitespace,
            backing up the last pointer. */
        while ((*last > *first) &&
            isspace(*((*last)-1)))
            (*last)--;
        /* restore null at end of string. */
        **last = '\0';
    }
    if ((side == 'l') || (side == 'b'))
        /* scan string forward for whitespace,
            bumping the first pointer. */
        while (**first && isspace(**first))
            (*first)++;
```


SINGLE BOARD COMPUTERS-6809

PART TWO - ST-2900 System

A short recap

Recently we received for review 3 different single board 6809 computers. All three are 64K systems, with 56K standard per FLEX™ convention. All three boards run FLEX, two have also licensed OS-9™ level one. The two FLEX syst s recommend that you purchase FLEX from your favorite source and use their modified drivers. Essentially this requires most any FLEX.COR and append the drivers to make a bootable FLEX system. Some consideration should be given to certain SWTPC FLEX versions, however, all can be made to work. Specifics will be covered in the review of each system.

The three systems we will look at are:

1. The PT-69™
   Peripheral Technology
   3760 Lower Roswell Rd.
   Marietta, GA 30067
   404/973-0042

2. ST-2900 System™
   Sardis Technologies
   2261 E. 11th Ave.
   Vancouver, B.C.,Canada V5N 1Z7

3. The 6809 "Unlboard"™
   Digital Research Computers (of Texas)
   P.O. Box 461565
   Garland, TX 75046
   214/271-3538

** (see footnote)

Notice should be taken that we will review each system in the order of A-Z. Why? Well they all have certain strengths and weaknesses, as we see it. Also we ended up having no particular favorite, as each has certain merits not available to the other two. All three are advertised in 68 Micro Journal and are running either in our offices or our lab (meaning they have been tested and accepted by our standards). All three perform well. Any one of the three when combined with disks and a CRT or keyboard and monitor (depending on the system) make an excellent, general purpose or specialized 6809 64K computer. The boards alone make great and very economical 6809 controllers or stand-alone systems. I see an upsurge in 6809 activity due to the economy and availability of these systems running all the popular 6809 disk systems and software!

I have long seen the need and attempted to get some of our present 6809 computer manufacturers to make a similar system. A very accurate survey some two years ago indicated that many of you wanted such a syst . Only SWTPC and WaveMate have done so.

WaveMate blew it by making the hardware and software dependent on a double density disk directory for FLEX. All normal FLEX systems use single density directories, for both single or double density format. Had they listened I sincerely believe that they would have had a winner, but now only SWTPC advertises a system in a desktop configuration (X-12+), and I understand it is doing quite well. However, by utilizing a CRT terminal similar to the Heath H-19, which has provisions for disks also, the entire system can be in one package. And that is the wave of the future, something we should have done years ago. Now with two of these, desktop complete systems are possible. With the other the size of the board is slightly too large, due to features not available on the other two. Remember, I said advantages and disadvantages.

Now, as to the Heath H-19, it is no longer in production, but many are advertised as used and at very good prices, so it should not be too difficult finding a low price used one or a similar type. Should any of you out there have a used Heath H-19 for sale, please let me know as I am certain I will be receiving many inquiries for availability of used ones.

SARDIS ST-2900 6809 Computer

As you might have noticed some of the remarks above are identical to last months review. As we are still growing and some articles in 68 Micro Journal are being reproduced elsewhere, I have decided to repeat a small portion so that each will be a complete review, as compared to the other two.

The Sardis Technologies ST-2900 system that we will review is a two board system. One board comprises the main 6809E CPU, 16/64K of RAM, provisions for 2716, 2732 or 2764 EPROMs, 2 RS232 serial ports, 16 bit counter timer, baud rate generation and other computer functions to be covered later.

The board is high quality glass epoxy, Eurocard size 3.9 X 6.3 inches, solder masked, silkscreened component overlaid, plated through with expansion socket. This is the smallest, most compact 64K board we have seen, however, it does not have the disk controller on the same board as does the other two.

The other board is the floppy disk controller card, with DD DS 1793 controller, 2 each 8 bit parallel ports, 2 each 16 bit counter timers and a prototype area (a darn good idea - if you have the space). These two boards mate together to form a very compact package, and with the prototype area are especially handy for that application requiring a 6809 64K computer system and some special circuit requirements. Not to mention a fine general purpose 6809 64K computer, which is what we have dedicated our ST-2900 system to.

THE CPU CARD

Sardis does not, according to the documentation we received, sell the boards fully populated. They are sold as bare boards or with all components installed, including sockets, less the more popular ICs. Ours came partially complete lacking only the more popular ICs.

For those 6809 users who are used to working with 'standard' Motorola I/O and other peripheral devices there will be required some device study here. As the I/O devices are not the 'normal' 6821 and 6850 ACIA or PIA interfaces. Instead the Signetics 2681 Dual Asynchronous Receiver/Transmitter (DUART) is used. For space considerations a wise choice, as it d s a lot more than the other devices. For ease of installation and interfacing it is a new device for many and requires some study of the included manufacturers spec booklet.

Also another serious consideration is that the DUART requires different initialization and addressing compared to the 6850 and 6821. Or, in other words, a lot of standard and normal FLEX and OS-9 software will have difficulty 'talking' to this system, due to the differences. The DUART is also addressed in a different memory area than most other 6809 system that follo the normal conventions (see memory map, later on).

An example of this is STYLO™. Stylo normally addresses the ACIA directly, bypassing the standard FLEX I/O routines. This is due to a difficulty with the input routines in the early versions of FLEX not have a 'no echo' input routine. Later versions do have this feature and pose no problem as STYLO has a flag (not documented, call Great Plains about your particular version) that allows the 'no echo' routine in FLEX to be used. Any other software (Dynacalc™, etc.) that talks to the port directly will have to be modified to work with this system.

It required only a two byte change to STYLO for us to get our FLEX 2.1 version running on this board, but I had to find the proper flag add esses. I would give them to you here but I have other versions of STYLO that either do not have this capability or it is located in a different place. I do not have the answers for any of the other software so cannot say. If your program does X-Y cursor addressing, you will probably have to make some program modifications. Don't forget to also reconfigure your program for a different CRT terminal, as we did, oh well.

The DUART

The DUART is a pretty potent fellow. The list of features is impressive. So much so that trying to single out the serial part, for instance, was a slight hassle because of all the other verbiage inbetween.

IN addition to two independent ACIA type devices it contains the following:

Quad buffered receive registers
Programmable data format
Programmable baud rates for each device
One user defined rate from timer/counter
Parity, framing and overrun error detection
False start bit detection
Line break detection and generation
Multi-function programmable 16 bit counter/timer
Multi-function 7 bit input port
Multi-function 8 bit output port
Interrupt system with 8 maskable conditions
Transfer rate - 1X-1MB/sec, 16X-125KB/sec
Auto wake up mode
Start-end break interrupt/status
Detect break within character
On chip Xtal osc
TTL compatible
Single 5v+ supply

As you can see, the choice of the DUART was not necessarily a bad choice, it is just that it make running a lot of 'standard' 6809 software a little (or a lot to some) more 'smarts' intense. For the controller or 'special' application, especially where space is a premium, then this is the system. And that is exactly where a lot of 6809 systems are now performing.

DOCUMENTATION

Again the documentation is sufficient, but not over bearing. Certainly not up to Heath kit standards. Actually, as a printer and publishe , I can tell you if the documentation for any of these systems were Heath quality the price would reflect an increase. For the average user, who has built kits or bare boards, it should be no sweat. Heck, I did it!

The schematic drawings and parts list, parts overlay diagrams, configuration charts and other data is completely sufficient. In fact the one page devoted to RS232 cable construction, is better than many I have seen in far more expensive systems. And that is one thing that must be kept in mind, these all are economy systems, but have very little if any compromise in quality or versatility. The board schematic is a double size single sheet, and if you have ever traced out a line then you know the value of single sheet diagrams. Actually it came in very handy for us as our system did not come up on the first try. It turned out to be a shorted bit 8 data line. We never did find the short (under some component) but we opened the foil and installed a short wire jumper and all worked as advertised.

Included in the CPU board documentation was the ever important memory map. You would be surprised at how many systems, over the years, I have seen that left it up to osmosis.

Memory Map

```
0000-feff  RAM
ff00-ff1e  I/O 0 (off board)
ff20-ff3f  2681 DUART (ff30-ff3f mirrors ff20-ff2f)
ff40-ff5f  I/O 2 (off board)
ff60-fff1  Various SAM registers
fff2-ffff  Interrupt vectors in EPROM
```

This mapping is for the system after the monitor has gone thru the init and power up routines.

The kit section of the manuals is fairly simple. The instructions are easy to follow and lead in an orderly manner. Our particular system was a very early one and the documentation has some penciled in notations. I would assume, and was told, that by now this has all been taken care of. However, the pencil notations were appreciated as they made it all the more simple. And SIMPLE is great!

An included detail and (again) simple section on 'starting up' could save you a lot of grief. Here I would suggest that you take your time and follow the instructions to the letter. The monitor has built in routines that check a partially populated board (minus many EXPENSIVE parts) for errors, and reports if all is o.k., if not beware, go back and recheck. A nice touch.

In addition to the regular parts the only one that would be difficult to find possibly (provided you purchased a board partially complete) are the 2681 DUART and SAM (Motorola stock part). We got a little help on these. The reminder of the parts are all common garden variety TTL devices, and assorted capacitors, resistors, etc.

Complete jumper options are mapped and explained, should be no problem there. Even where trim pots are required, a nice small feature is included. The board is laid out and drilled to accept either square or long type pots, so whatever you happen to have will do.

All in all this set of documentation is at least on a par with the others, and in some small ways a little easier to use.

The FDC Floppy Disk Controller

The floppy disk controller board is the same size as the CPU board. They fit together, back to back by a 60 pin connector. Once installed they make a very compact unit.

The controller is a 1793, a standard device. The only device on this board that is somewhat strange to some is the VIA (Versatile Interface Adapter) 6522. No spec sheet on the VIA was included, however, it is a fairly common part and should pose no difficulty.

All of the comments above conce ning the CPU card and documentation apply also to this board and we had no

problems at all. Again a short 'first time' power up routine is given - follow the instructions. Trim pot alignment is a simple measurment of two different voltages and took only about 30 seconds or so. DD DS was no problem. No disk errors or other related problems.

Although only 40 track drives have been used with this system, it should handle 80 tracks as well. Also I was told that more documentation has been finished and that only means that things should be (that word again) simpler.

Total construction and installation time into a Heath H-19 CRT terminal was about 6 hours. About two of these were trouble shooting the CPU card for a shorted data line, between answering numerous telephone calls and transac ing other normal daily routines.

FLEX Conversion Package

Here again the manufacturer lets you go out and dig up your c y of FLEX. Not all his fault as TSC has about priced General FLEX out of reach of most ($250.00). However, again I can only suggest you purchase FLEX from your favorite FLEX dealer. Call us if you can't find one.

What you receive with the FLEX conversion, newdisk, printer drivers, etc. routines is sufficient to develope a bootable FLEX disk. However, due to the memory mapping of I/O devices it is somewhat different than most, but no more difficult. We followed the instructions that came with the development disk (price unknown) and had a FLEX disk running in about 5 minutes. After that ran just like all other FLEX operations.

I was told that OS-9 may be available. However, at this writing, to the best of my knowledge, only the other two offer OS-9 level one. If interested write and ask.

One nice feature of the FLEX newdisk furnished (no source - why?), is hat you can specify either narrow or wide gaps. Normally the IBM standard for 8" disk is wider gaps, less sectors, the TSC system calls for shorter gaps and more, or extended sectors. I have never seen a 'IBM standard' for 5" disk.

Also included is a program called DSKSET. DSKSET can be called from STARTUP or as a command with passed parameters. It sets up nearly any parameter for the disk drives needed, from seek speeds, single/double density, single/double sided, track densi y, max tracks allowed per drive, media track density, sectors per side, read error count, write error count and verify error count, so name your precision or poison.

ST-MON

The monitor, ST-MON has all the normal monitor calls or routines, except breakpoints and register routines. However, it does a memory test on power up and some other functions not normal to some monitors. As configured it is set up for 9600 baud for the communications port. Should you desire another rate a new monitor EPROM should be blown - only the first byte (byte 0) need be changed - a table is given.

PRICE

The price should be taken from their latest advertising in 68 Micro Journal, as the ones we have expired in August.

With the cost of the ST-2900 System, a used CRT terminal and two 5" disk drives DD DS, the total system cost should be $1500 or less, and that is the advantage of the new wave of single board 6809 computers, cost, compactness and semi-portability (with the Heath terminal the total weight is about 45 pounds).

For those applications demanding additional I/O, hard-disk and other peripheral interfacing then one of the larger S50 Bus system will be required, but for many this is the way to go. And it is good for the industry, for experience has shown that satisfied small system owners eventually graduate to larger and more complex 6809 systems rather than go off to the 'other side' (who wants to learn new languages, buy new software and essentially start all over?)

Next month a review of the 6809 Uniboard from Digit Research Computer (of Texas).

- - -

** Since we started this project another new 6809 single board computer has come to light, see MICROKEY Ltd advertising this issue.

I should have a review of the MICROKEY 4500 6809 FLEX computer System by the time this series finishes. As you can see from the advertised specs, this single board 6809 computer, like the others, is unlike the others. Each has certain very important differences, which leaves them complimenting each other.

So now there are four, and I understand that a revised and improved version of one previously advertised will be back. Could there be 5 or more? - stay tuned-in.

DMW

- - -

Computer Excellence Memory Board

Some time ago I received a 256K memory board from Computer Excellence for review. This board contains 256K of dynamic RAM with completely hidden refresh. It will run at 2 MHz. The board also has a self contained DAT. Because of this, it will work in older systems that do not have the extended addressing of the SS-50C bus implemented. Since my original system has never had the extended addressing implemented, I was eager to see how it would work.

As you might suspect, the hardware configurations with which this board has to work are numerous. My system has a SWTPc MP-09B processor, a GIMIX DMA disk controller, and a pair of 8" disk drives. I have never implemented extended addressing, since I had (until now) 56K of memory, and no reason to go beyond that limit.

Using the memory board DAT has one limitation. There must not be other memory in the system, which, without extended addressing, would appear redundantly throughout the memory map of the 256K b rd. In order to use the C.E. board in this configuration, it is necessary to do two modifications to the processor board. First the processor board DAT must be disabled by jumpering around the DAT circuitry with the four high order address bits. If your processor board has the DAT chip in a socket, you can plug a jumper header in place of the chip. Secondly, the address decoding must be modified so that when the processor writes to the DAT address, it will write to the bus and not to the internal processor DAT register, (or perhaps in addition to the internal DAT register). These modifications are relatively straightforward and will be documented in the manual.

Having done those, I set the configuration switches on the C.E. board to the proper positions and found the system to work. I had been running my system at 1 MHz because I had one of the old Motorola built SWT c 32K dynamic RAM boards which won't run any faster than 1 MHz. I swapped crystals in my processor board, which by some forethought was a 2 MHz version to begin with, and found that I could now run reliably at 2 MHz.

What on earth would I want with 256K? Actually because of address conflicts at $ED00 with I/O, and at $F000 with the disk controller and monitor, there is just 248K available. The main use at present, is "virtual disk". The board comes with a disk containing software that makes it useable for this. There are several utilities supplied. Before running any of them, you must edit a short file called CHANGE-ME which is used by all the utilities to configure them for your system. You must supply only a very few system constants in CHANGE-ME. After doing this you may assemble and run the utilities.

The first is called TSIZE, and it displays a memory map of the available memory with the new board in place. It indicates 248K available on the first four "pages" of extended memory. The system blocks at $E000 and $F000 are indicated as I/O and Monitor respectively.

The second utility is called TURBO, and it is the utility that "formats" the remainder of extended memory (pages 1,2, and 3) as a disk. It reports 752 sectors, and from that point on, you may use "Drive 3" just as though it were a physical disk drive. The utility TSET allows you to make Drive 3 both the System and Working drive. Now you can copy your utilities or your favorite compiler and editor to Drive 3, and use it to edit a file and compile it, for example. After you are done, you can COPY the resulting file back to a "real disk". Running my system at 2 MHz. resulted in a speed increase while

using the 8" real disks, after I reformatted some disks to take advantage of the GIMIX choice of optimum sector interleave for the 2 MHz system. Even with the higher speed, I found that using the virtual disk I was able to load and save files in less than half the time required to read and write to real disks.

If it were only for this one use of the extended memory, the C.E. card would be worth the price. The board is set up for "task switching" as well. While I have not tried it in this mode, it appears to be quite straightforward to provide a single user with the capability to leave some task with files open, editors in memory, etc., and go to another task with different files open. Then the user can switch between these tasks at will!

The board will soon be (or is now) available in configurations with 512K and 1 Mbyte, thanks to the availability of more dense dynamic RAM chips.

The system works quite well with any of the programmed I/O disk interfaces such as the DC-X series of SWTPC and the GIMIX 28 and 58 boards. There was what we thought to be a hardware problem with running the memory board in a system with a SWTPc DMAF-2 DMA disk controller. It turns out that SWTPc has done something to the versions of FLEX that they supply for running with the DMAF to make it use extended address page $01 for the first 48K of memory, i.e. the user memory from 0000 to BFFF. Then it switches and uses extended address page $00 for FLEX itself. This fouls up the loading of the DAT by the software supplied with the board.

It is possible with these versions of FLEX to specify that the virtual disk will start using memory on extended page $02, but this reduces the amount of memory available for virtual disk so that there is "only" 128k of virtual disk (492 sectors) available rather than the 752 sectors with the programmed I/O or GIMIX DMA versions of FLEX. There are a couple of possible solutions. The first one is to find the place in these versions of FLEX where the DAT is being reloaded with the extended page $01 addresses and disable that. C.E. is working on that solution now and will probably have it done by the time you read this. The second is less desirable, and it is to supply a DAT table for the virtual disk software that will be compatible with these versions of FLEX.

So much for the hardware and software. What about the manual? The manual contains a well written though lengthy discussion of the principles behind memory management (or the DAT) schemes that allow a 16 address line processor to use 1 Mbyte of memory. The manual then goes into the configuration of the board for all the possible combinations of Processor boards, Disk controllers, Extended memory, Non-extended memory, etc. Rather than confuse the user with so much information, it would be better to present the information in tabular form. The user could then find the combination of parameters that describe his system, and read the required switch settings directly from the table. I have to say that the apparent complexity of configuring the board did put me off in my testing for some time. I am happy to report that the software configuration was simplified considerably in the process of my working with Computer Excellence on getting the board and software configured for my system(s). I was able to test some configurations that were not available to C.E. and that is how we discovered the problem with the DMAF board compatibility.

I've spoken to the folks at Computer Excellence regarding the manual and the apparent complexity of configuring the board, and suggested putting a configuration table in the front of the manual and placing the technical discussion in an appendix at the rear. They have promised me a copy of the revised manual as soon as it is done. They have been selling the manual separately, and they had wondered why no one who bought the manual had purchased the board. They seemed to be pleased with my observation that all the technical discussion had probably frightened off the possible customers, and agreed immediately with my suggestion of a simplified installation instruction section at the front of the manual. Most of us, when we buy something like this, want first of all, to get it working. Then when we get it going, we can explore the technical details at our leisure.

The manual includes a great deal of information, including a full schematic of the board. I know firsthand that Computer Excellence is willing to work with customers who have difficulties configuring the board, the software, or their systems compatibly. Our SS-50 systems are now so diverse with respect to configuration and hardware, that no one software supplier has "one of each" on which to try his software. Running Dynamic RAM at 2 MHz with "transparent refresh" is no simple trick. The folks at Computer Excellence are to be congratulated on this fine product.

Computer Excellence Inc
P.O.BOX 8442
Coral Springs, FL 33065
1 (305) 752-8321

256K memory board: $749.00
512K memory board: $1495.00
1 Mbyte board: $2495.00
** Memory Prices subject to change **

Review by Ron Anderson

- - -

* Editor's Note: The 68 Micro Journal labs have practically all available (and some not available) 68XX computer systems, up and running, or available in stock for testing purposes.

Perhaps I should make some of you manufacturers of hardware and software aware that we can test your product on practically ANY 68XX(X) system, available to our user community. Please call, if interested, for details including rates.

Might as well throw this in here also - we do manuals and consulting about your product - viability - market needs and responses - suitability - testing - research, etc. All of this is done through our **DATA-COMP** Division and some of the MOST successful S50 Bus vendors use our services on a regular basis - nuff said?

DMW

- - -

68000 USER NOTES

Philip Lucido
2320 Saratoga Drive
Sharpville, PA 16150

Struck Down by Gremlins!

Some of you may have noticed that my column did not appear in the October issue, while the November column, in large part, consisted of mysterious references to that missing column. I'm not sure exactly what happened, but anyway, the following should have been the October column, and as such precedes the November in its concerns. Since so many references were made then to errors in this column, I've decided to leave all of those bugs alone. Read this one, then go back and read November's again. Hopefully, things will be less confusing.

• • •

Ed's Note: Sorry, but Uncle didn't deliver as promised. They (articles) are processed as received, but one never arrived, this one. Thanks Phil, not your fault. We appreciate your effort. Ours and the PO.

DMW

• • •

This month, the focus is on OS-9 and 68000 system software. A quick aside - I'm getting tired of typing OS-9/68K. From now on, assume that OS-9 refers to the 68000 version. I'll call the 6809 version OS-9/6809.

OS-9 Version 1.0

I have been expecting version 0.6 for some time, but it turns out that Microware shipped it, towards the end of June, for only a couple of weeks. Instead, they have been working hard on getting version 1.0, presumably the first non-preliminary version, ready for shipment.

I don't have V1.0 yet, but was on the phone with Microware today, and was told a little of what to expect. First of all, the module format will be different, so any current programs will need to be recompiled under the new version. There will also be a few more utilities included with the package, and bugs found in the preliminary versions will be repaired. A new version of the C compiler, as well as the first version of Basic09, will be out at the same time.

The screen editor included with OS-9, scred, required the C compiler to configure it for a particular terminal under version 0.5. The new scred, however, will not require recompiling for most terminals. Instead, a package of configuration files is installed in the SYS directory. Also, scred will be greatly improved. The preliminary version was fairly 'buggy'.

The best news has to do with keeping the size of programs down. Almost all of the utilities in OS-9 are written in C, and are very large when compared to 6809 versions. Much of the problem is not due to any gross inefficiency on the part of the C compiler, but to the number of modules which are linked into the code. For instance, the formatted print routine in C, called printf, is used in just about every program. Printf works with the buffered I/O routines in the standard C library, which means that it requires a large number of support routines to be linked in with it. Thus, a quite small program, which by itself might require only 300 or 400 bytes of code, will suddenly need 3000 bytes long after linking.

Under OS-9 version 1.0, a much better method will be available. Instead of each utility needing to link in the same routines, these routines from the C library will all be collected in a number of sharable system modules, which will be loaded once, at startup, and kept in memory from then on. Any utility requiring one of the routines has only to link to the proper module. The physical linkage to the library routines will be provided by the 68000 TRAP instruction. With these central modules, the only code actually linked into each utility is a very short routine which sets up the proper parameters and does the TRAP.

Three different system library modules will be available. CIO will be about 8K long, and contains printf and scanf, the formatted output and input routines, as well as the various buffered I/O routines like getc and putc. MATH1, about 4K long, provides single and double precision floating point operations, ASCII to floating and floating to ASCII conversions, and those 32 bit integer operations which are still fairly long in the 68000, like 32 bit by 32 bit multiply and divide. Finally, MATH2, about 6K long, provides transcendental operations, which include trigonometric routines like sine and cosine, and various other mathematical functions like e^x and log(x).

The central shared modules should cut the size of utilities way down, to maybe a half or a third of the current sizes. But the savings in code size is not the only benefit. Since the library routines are not actually linked into a program, a change in a library function will not require all programs using that function to be recompiled. Instead, only the system module needs to be changed. As a dramatic example, consider a system in which floating point hardware is newly installed. Under just about any other operating system I know of, all existing applications would have to be recompiled with a new library for floating point support. Under OS-9, the applications don't change at all! Instead, the MATH1 and MATH2 system modules are replaced with newer versions, and the job is complete. What could be simpler?

These system modules are obviously available for any program which requires their capabilities, not just C utilities. For instance, Basic09 does all of its floating point operations with the modules. All in all, the shared library module looks to be a terrific idea. The only drawback is the extra run time required for the TRAP to execute. When compared with the time required to do I/O or floating point calculations, though, this becomes insignificant. By the way, Microware says that you are not locked into using these routines. The complete routines can still be linked into your code, as an option.

I Get Mail!

In my first column, I mentioned that my copy of OS-9 was the second one sold. I got a letter from Kirk Anderson, the purchaser of the first copy, who described his experiences with the new system. First off, Kirk's computer has two 64K memory boards. His motherboard did not decode address lines A16 to A19, though, so the I/O ports appear in each 64K page, in effect giving him a machine with two non-contiguous blocks of 56K RAM each. OS-9 can be brought up in such a system, but certain utilities, such as r68, the assembler, look for 64K or more in a contiguous block when run, at least in preliminary versions. The solution was to modify the motherboard so the I/O ports would appear in only one block. However, this is for a Level 1 system. When Level 2 is available, the mapping of non-contiguous RAM into a contiguous logical block should make the motherboard modification unnecessary.

This raises the question of just how much memory you should have to run OS-9. Microware believes that 128K is a minimum figure. The kernel, plus a typical mix of I/O drivers and managers, requires about 26K of memory. On top of this, the shell requires 14K for itself. Thus, you need 40K of memory just to see your first shell prompt. Add to this the generally large data areas used by various programs, and 64K is simply not enough. My own system has 256K, and I have never had any problems caused by insufficient memory.

What's Wrong With '#'?

Kirk also had a question concerning the '#' command line modifier, which increases the initial data area size. When used with the various utility programs, like copy, this modifier seems to have no effect at all. How do you specify a memory qualifier, then?

The '#' *actually* does increase the data size, as it should. The problem is not with the OS-9 shell or kernel, but with the way the utilities were written under C. In a C program, the data area is divided up into sections. Global and static variables are stored at the low end of the data space. At the upper end are the command line arguments and the stack, which grows downward toward the variables. Between the two blocks, in the middle of the data space, is a buffer area which is used for I/O buffers by getc and putc. It is this middle buffer which will be increased in size by a '#' modifier. It is available for access using C routines freemem() and lbrk(), which return the size of the block and allocate memory from the block.

When using that middle block, unfortunately, you don't have a firm handle on the amount of RAM available, since room must be left for the stack to expand. A simpler alternative is to allocate memory above the stack, which is in no danger of being invaded. This is done by a routine called sbrk(). Since the memory above the stack was not part of the initial allocation, sbrk() runs by requesting more memory from the kernel, via an OS9 F$Mem assembler call.

The '#' modifier does not affect the OS-9 utilities because they use sbrk() calls to allocate their buffers, instead of lbrk(). To allow you to specify a size for their buffers, they use an option, -b, instead of following the '#' allocation. For instance, to set a 20K buffer for a copy, use 'copy"-b=20K' instead of 'copy"#20K'.

Personally, I wish the utilities would use the lbrk() middle buffer instead. For one thing, the -b option requires coding within the utility, while the '#' modifier is handled directly by the shell. More importantly, I think there is a possible problem with calling sbrk() in a Level 1 system. Under Level 1, all data area RAM for a single task must be contiguous. This is generally no problem for a single user running one task at a time, but if a number of routines are run concurrently, say in a pipeline, a program might try to request more memory, only to find that the extra memory above its current data area end is already being used.

***** NOTE - see the November column for an update on this - Phil *****

However You Look At It, Fast Is Fast!

Last month, I reported on the speed of the 68008 in a benchmark which did very little I/O, so its speed was determined by the speed of the memory and the 68008, not by the speed of the disk or other I/O. Such a program is known as computation-bound, and a faster processor should translate into a faster program. As expected, the 68008 did indeed run faster than the 6809. This month, in the course of writing some general utilities, I ran a program which sorted a file of 127000 characters, organized as 24000 words, one per line, into alphabetic order. The program did so by creating a large number of temporary files, each with a small section of the text, and then merging the temporaries. This would appear to be a program that is I/O-bound, and since the 6809 and 68000 versions both run on the same disk, sorting the same file, I was not expecting any great difference in speed between the two processors.

To my astonishment, the 68008 took far less than half the time of the 6809. The 6809 version came in at 27 minutes, while the 68 8 took only 10 and a half! Frankly, I'm still trying to figure this one out. Perhaps OS-9/68K uses larger buffers, or maybe Microware has greatly increased the efficiency of disk handling. Another possibility may be that the 68000 uses Level 1 OS-9, which has no memory mapping, while the 6809 uses

Level 2, which uses relatively slow memory bank to bank copying during disk I/O. Nevertheless, this can hardly account for a 17 minute improvement.

Whatever the reason, I am definitely impressed. So, it seems, is Microware. In calls to them, I learned that they have been quite surprised by the speed of OS-9/68K on what is in effect an 8 bit chip. On paper, a 68008 doesn't look that much better than a 6809. In a real machine, something entirely different is happening. Part of the reason is doubtless a 68000's ability to hold a lot more data in its registers at any time, reducing memory references. Also, this OS-9 is the second for Microware, and all the experience on the 6809 may be showing up on the 68000. ***** Again - See November for more. *****

I May Never Assemble Again

It's time to bring up that old bugaboo, assembly language (AL) versus high level language (HLL) and compilation. I am hesitant to do so, seeing the difficulty that Ron Anderson has been having in stopping his debate, once it was started, but there is just no way around it.

Under the 6809, the burden has been on compilers to prove themselves, in terms of code size and efficiency. The situation in regards to the 68000 is reversed. As I've noted throughout my columns, almost everything is now written in an HLL, in particular the language C. It nows seems up to supporters of AL to back their position, not the other way around. What gives?

I think what happened was a change in emphasis. As we progress from generation to generation of microprocessor, the yardsticks of performance change. With the 6809, the best programs were those that performed a task using the fewest bytes or clock cycles. This made some sense, as the limits of 8 bit machines, with 64K address spaces and limited machine registers, put space and time at a premium.

With the 68000, there is address space to spare, and a powerful enough instruction and register set to speed things up significantly. Code can afford to be a little looser, because the limits governing that code have been loosened. Still, all this extra power does not come without a cost. It is no longer enough for a program to be fast. It should now be ever more powerful than previous versions. After all, what's all that RAM for, if not added features?

If there's one thing a good HLL can do, it's allow a programmer to keep control over a program as it grows in size and complexity. Structured programming is possible with AL, but the emphasis on detail, keeping track of registers and the like, tends to obscure the 'big picture'.

Of course, the right HLL can make all the difference, and it's not surprising that C has become preeminent in the 68000 field. One of the things AL has going for it is its flexibility, something that is missing from some HLLs. However, C is nothing if not flexible. For instance, few HLLs allow subroutines to be called with a variable number of parameters, but this feature is practically central to C, being used to great effect in printf, the formatted print routine. According to some, C is more of a medium level language, sitting somewhere between AL and HLL. C does have the feel of AL at times, with its almost universal reliance of pointers, and the ability to control register assignment of variables, but it still goes together like an HLL.

Some examples might help. While writing these columns, I have had to create a few utility routines. One of these is so trivial as to almost be a throwaway. To count words in the text files, I first need to filter out the command lines, which are any lines starting with a comma. For simplicity, I decided to write a program which read from the standard input, and wrote to the standard output. This would not be a difficult program in AL, but anytime I sit down to write a program in AL, I end up taking a minimum of 20 to 30 minutes, even more with a new operating system with which I am not completely familiar. In C, writing the program took only 5 minutes, with 2 or 3 minutes spent on the compilation. It is only 29 lines long, including 11 lines that are blank or comments. The object code may be longer than that for the same program in AL, but so what? It's my time that is important here, not the computer's RAM.

68MJ would prefer that I send these programs in on Flex format 5 inch disks, but the word processor I'm using operates under OS-9/6809, and due to some technical problems, the O-F program for transferring data between Flex and OS-9 doesn't work on my system. Anyway, since my hard disk has segments devoted to OS-9, and others devoted to Flex, why not write a program to transfer directly between these segments, avoiding floppy disks entirely? I end up writing two programs running under OS-9, one to take a directory of a Flex segment, and one to copy from Flex to OS-9 and back again. These two programs are quite large, and have to deal directly with such Flex constructs as the directory, the free chain, and the system information record. Still, they were written in the space of a few weeks, in my spare time, as I was busy writing other programs.

Finally, there is what may be the most telling reason for using C over AL. The utilities I just described were in fact written under OS-9/6809, even though this column is for the 68000. The editor I'm now using isn't available under OS-9/68K yet, so I can't entirely abandon the 6809. But with programs written in C, it doesn't make a lot of difference which operating system I use. For the past month, I have been writing C programs which compile, without modification, on both the 6809 and the 68000. The 68000 versions tend to be a litte longer and a little (or a lot) faster, but the programs run the same way.

So is AL dead on the 68000? Of course not. Some programs need to run as fast as possible, such as the kernel in OS-9 or the reportedly very tight graphics routines in the Macintosh. I have written some AL programs myself for the 68000, some just for the experience, and some because the function they performed was at such a low level that C couldn't handle it. Even if a program is written in C, writing some subroutines in AL to speed the program will make some sense. But from now on, it seems to me that the default language of choice for the 68000 is some HLL, usually C, instead of AL.

The Management Welcomes Opposing Viewpoints

I may be asking for trouble, but if you have any views on the subject, send me a note. I will not be able to devote quite as much time to the subject as the Flex column has, but I will let other people put in their two bits' worth.

Back next month, when I should have version 1.0 of OS-9. I will also set forth a plea for some standards in the user interface in programs, and maybe print some of my own tool programs as an example.

TURTLE GRAPHICS IN PL/9

Several years ago, when I was using a 6800 system, I was introduced to Pascal. To aid in the learning process, my dad bought the book PROBLEM SOLVING USING PASCAL by K. L. Bowles. This book was obviously meant to be used on a UCSD Pascal system, and most of the beginning chapters used the turtle graphics that come with that system. Well, I learned Pascal, but I still could never try out those neat pictures, so I set down one weekend and wrote a turtle graphics program that would print the pictures on my Epson printer.

Now, all of that was several years ago, and since then the system I'm using has been changed to a 6809, and no new Pascal was purchased. However, my dad bought Windrush's PL/9 compiler, and I set about to convert my graphics driver. The following routines are that result.

Turtle graphics, as gleaned from PROBLEM SOLVING USING PASCAL are as follows. There are 3 types of commands: movement, turns, and color. The movement commands are MOVE, and MOVETO. MOVE moves x units in the dir tion the turtle is facing. MOVETO moves the turtle to the point x,y. The turn commands are TURN and TURNTO. TURN turns the turtle x degrees counter-clockwise from its current heading. TURNTO turns the turtle to the value of x degrees. PENCOLOR is the only color statement, and the colors may be WHITE, BLACK, or NONE.

The screen the turtle moves on is 480 pixels by 480 pixels with the origin (0,0) in the center. X and Y values may be between -239 and 240. The colors the program refers to are actually not what are plotted on the printer. Using the analogy of a digital plotter, the color WHITE would put down a pen, the color BLACK would put down an eraser, and the color NONE would lift the pen.

The drawing is stored in memory in a format that can be directly sent to an Epson printer equipped with GRAFTRAX or GRAFTRAX+. Each byte of the picture holds eight dots of the picture arranged vertically. These vertical lines are then arranged in horizontal bands. This may sound strange at first, but the print head has its pins arranged vertically and goes across the paper horizontally, so it works nevertheless.

Along with the listing of TURTLE, I have included a small demo program called SPIRAL. It computes the design most commonly associated with turtle graphics. Use this program to test TURTLE after you load it into your computer. If you get a nice spiral design, as shown below, great! Incidentally, line numbers 5, 7, and 10-11 of SPIRAL should be included in every file using TURTLE.

Modifications to TURTLE should hopefully not be too difficult. PL/9 is a structured language and has most common loop structures. TURTLE requires SIN and COS to properly compute the MOVE routine, so I include the library file SCIPACK. In addition, two print routines, PINIT and POUT, are included in TURTLE, but you may want to put them in your own IOSUBS library file, which is a better place. If you do not have enough memory to use the size picture I chose you will have to modify the size of the picture (x * y / 8) as well as the constant definitions in TURTLE. If the program still does not work, the best thing to do is study the program. I think I have included enough comments to make it understandable, but there are never enough comments.

Have fun making designs!!!!

Steve Cole
5868 Pentz Way
San Jose, CA 95123

```
/* TURTLE GRAPHICS LIBRARY FILE FOR EPSON PRINTERS OR ANY
PRINTER CAPABLE OF BIT GRAPHICS. THIS PROGRAM IS WRITTEN IN
PL/9, BUT SHOULD BE EASY TO CONVERT TO ANY OTHER STRUCTURED
LANGUAGE. THIS PROGRAM STORES THE PICTURE TO BE PRINTED IN
MEMORY, ONE BIT PER PIXEL. THE PIXELS ARE STORED IN AN
UPRIGHT FASHION, THE WAY THE PINS ARE ARRANGED ON THE PRINT
HEAD. IF YOUR OWN PRINTER DOES GRAPHICS SOMEWHAT DIFFERENTLY,
YOU'LL HAVE TO CHANGE THE WAY THINGS GET STORED.
*/

CONSTANT NONE=-1, BLACK=0, WHITE=1,     /* THESE ARE THE
                                           COLORS USED BY
                                           PENCOLOR */
         XMAX=240, XMIN=-239,           /* THESE ARE THE MAXIMUM AND */
         YMAX=240, YMIN=-239;           /* MINIMUM VALUES OF X AND Y */

PROCEDURE PINIT;
   CALL $CBC0;
ENDPROC;

PROCEDURE POUT(BYTE CHAR);
   ACCA=CHAR;
   CALL $CCE0;
ENDPROC;

/* THIS PROCEDURE SIMPLY FINDS THE ABSOLUTE VALUE OF AN INTEGER. */

PROCEDURE ABS(INTEGER VALUE);
   IF VALUE<0 THEN VALUE=-VALUE;
ENDPROC INTEGER VALUE;

/* THIS PROCEDURE USES A QUICK METHOD TO FIND THE VALUE OF 2^I. */

PROCEDURE TWO_TO_I(BYTE I);
   BYTE LOOP,TTI;
   LOOP=1; TTI=1;
   WHILE LOOP<=I BEGIN
      TTI=SHIFT(TTI,1);       /* THE FUNCTION SHIFT(I,C) SHIFTS I C PLACES.
                                 HERE IT IS EQUIV. TO I*2. */
      LOOP=LOOP+1;
   END;
ENDPROC TTI;

/* THE COMMAND "PLOT" PUTS A PIXEL OF THE COLOR SPECIFIED INTO THE
PICTURE. THE FORMAT IS:

         PLOT (X,Y,COLOR);

WHERE X IS THE X POSITION, Y IS THE Y POSITION, AND COLOR IS THE COLOR.
THE COLORS ARE BASED ON THE UCSD TURTLE GRAPHICS, SO BLACK ERASES A COLOR,
AND WHITE IS THE DRAWING COLOR (USUALLY BLACK).
*/
PROCEDURE PLOT(INTEGER X,Y: BYTE COL);
   BYTE MASK,TEMP: INTEGER Y2;
   IF X>XMAX THEN RETURN;       /* THESE LINES SKIP PIXELS THAT WOULD */
   IF X<XMIN THEN RETURN;       /* NOT NORMALLY BE ON THE SCREEN */
   IF Y>YMAX THEN RETURN;
   IF Y<YMIN THEN RETURN;
   IF COL=NONE THEN RETURN;
   Y=YMAX-Y; X=X-XMIN;          /* PUT THE ORIGIN AT TOP LEFT */
   Y2=SHIFT(Y,-3); /* DIVIDE BY 8 */
   MASK=7-Y+SHIFT(Y2,3); /*  THIS MAKES THE BIT NUMBER */
   TEMP=PAGE(Y2*480+X);
   IF COL=WHITE THEN TEMP=TEMP OR TWO_TO_I(MASK);
   IF COL=BLACK THEN TEMP=TEMP AND 255-TWO_TO_I(MASK);
   PAGE(Y2*480+X)=TEMP;
ENDPROC;

/*
THIS ROUTINE PRINTS THE SCREEN OUT ONTO THE PRINTER.
*/

PROCEDURE PRINTSCRN;
   INTEGER LOOP,X;
   PINIT;           /* INITALIZE THE PRINTER */
   LOOP=0;
   POUT(27); POUT('A'); POUT(8);        /* SET LINE SPACING TO 8 PIXELS HIGH */
   REPEAT
      POUT(27); POUT('K'); POUT($E0); POUT($01); /* PRINT 480 GRAPHICS CHARS*/
      X=0;
      REPEAT
         POUT(PAGE(LOOP*480+X));
         X=X+1;
      UNTIL X>479;
      POUT (CR); POUT(LF);
      LOOP=LOOP+1;
   UNTIL LOOP>YMAX/4;
ENDPROC;

/*
THIS PROCEDURE CLEARS OUT THE MEMORY WHERE THE PICTURE IS SUPPOSED TO GO.
*/

PROCEDURE CLS;
   INTEGER LOOP;
   LOOP=0;
   REPEAT
      PAGE(LOOP)=0;
      LOOP=LOOP+1;
   UNTIL LOOP>28799
```

```
   XPOS=0; YPOS=0; ANGLE=0; COLOR=NONE;
ENDPROC;

/*
THIS PROCEDURE SETS THE DRAWING COLOR. THE FORMAT IS:
        PENCOLOR (COLOR)
WHERE COLOR IS WHITE, BLACK, OR NONE.

IF YOU TAKE A LOOK AT THE CODE, YOU WILL REALIZE THAT NO PROCEDURE
FOR THIS IS REALLY NECESSARY. IT IS INCLUDED ONLY TO MIRROR THE REAL
TURTLE GRAPHICS.
*/

PROCEDURE PENCOLOR(BYTE COL);
   COLOR=COL;
ENDPROC;

/* THIS PROCEDURE POSITIONS THE TURTLE AT A GIVEN ANGLE. THE FORMAT IS:
        TURNTO (ANGLE);
WHERE ANGLE IS FROM 0 TO 359. */

PROCEDURE TURNTO(INTEGER ANG);
   ANGLE=ANG;
ENDPROC;

/* THIS PROCEDURE TURNS A SPECIFIED AMOUNT FROM THE CURRENT POSITION.
THE FORMAT IS:
        TURN (+/- AMOUNT);
WHERE AMOUNT IS ANY NUMBER, POSITIVE OR NEGATIVE. */

PROCEDURE TURN(INTEGER ANG);
   ANGLE=ANGLE+ANG;
   WHILE ANGLE)=360 ANGLE=ANGLE-360;
   WHILE ANGLE<0 ANGLE=ANGLE+360;
ENDPROC;

/*
THIS PROCEDURE MOVES THE TURTLE FROM THE CURRENT POSITION TO THE SPECIFIED
POSITION, DRAWING A LINE IF THE CURRENT COLOR IS WHITE OR BLACK. THE
FORMAT IS:
        MOVETO (X,Y);
WHERE X AND Y ARE THE NEW COORDINATES.
THE VARIABLES XPOS, AND YPOS ARE UPDATED AT THE CLOSE OF THE PROCEDURE SO
AS TO CHANGE THE CURRENT POSITION.
*/

PROCEDURE MOVETO(INTEGER NEWX,NEWY);
   INTEGER X,Y,R,DX,DY,SX,SY;

/* XPOS, YPOS are the current positions at entry and again at exit */

   SX=NEWX-XPOS;
   DX=ABS(SX);
   IF SX<0 THEN SX=-1;
   ELSE SX=1;
   SY=NEWY-YPOS;
   DY=ABS(SY);
   IF SY<0 THEN SY=-1;
   ELSE SY=1;
   R=0;
   X=XPOS;
   Y=YPOS;

   PLOT(X,Y,COLOR);  /* THE INITIAL POINT */

   IF DX >= DY THEN
      BEGIN
         WHILE X <> NEWX
            BEGIN
               X=X+SX;
               R=R+DY;
               IF R >= (DX-R) THEN
                  BEGIN
                     Y=Y+SY;
                     R=R-DX;
                  END;
               PLOT(X,Y,COLOR);
            END;
      END;
   ELSE
      BEGIN
         WHILE Y <> NEWY
            BEGIN
               Y=Y+SY;
               R=R+DX;
               IF R >= (DY-R) THEN
                  BEGIN
                     X=X+SX;
                     R=R-DY;
                  END;
               PLOT(X,Y,COLOR);
            END;
      END;
   XPOS=NEWX;
   YPOS=NEWY;
ENDPROC;

/* THE FOLLOWING NUMBER IS USED TO CONVERT DEGREES INTO RADIANS */
REAL PIBY180 0.0174532925;

/*
THIS NEXT PROCEDURE MOVES FROM THE CURRENT CURSOR POSITION A SPECIFIED
DISTANCE IN THE DIRECTION CURRENTLY POINTED TOWARD BY THE TURTLE.
THE FORMAT IS:
        MOVE (DISTANCE);
WHERE DISTANCE IS THE NUMBER OF UNITS TO BE MOVED.
*/

PROCEDURE MOVE(INTEGER DIST);
   MOVETO(FIX(COS(ANGLE*PIBY180)*DIST+XPOS),FIX(SIN(ANGLE*PIBY180)*DIST+YPOS));
ENDPROC;

/* SPIRAL.PL9 */
/* DRAWS A PRETTY SPIRAL DESIGN */

ORIGIN=$9000; STACK=*;

GLOBAL BYTE PAGE(28800),COLOR,      /* LINES 5, 7, AND 10-12 SHOULD BE IN   */
           STR(20);                 /* EVERY PROGRAM THAT USES THE TURTLE */
       INTEGER XPOS,YPOS,ANGLE,     /* ROUTINES. */
           DISTANCE,ANGL,CHANGE;

INCLUDE SCIPACK;                    /* FOR SIN AND COS */
INCLUDE TURTLE;

PROCEDURE NEXTLINE;
   MOVE(DISTANCE);
   TURN(ANGL);
   DISTANCE=DISTANCE+CHANGE;
ENDPROC;

PROCEDURE WHILEPLOT;
   CLS;
   PENCOLOR(WHITE);
   DISTANCE=4;
   ANGL=89;
   CHANGE=1;
   WHILE DISTANCE<400 NEXTLINE;
   PRINTSCRN;
```

SOFTWARE PRODUCT REVIEW

Introduction

I will review some useful and cost-effective software, the cross-assembler macro sets by Computer Systems Consultants, Inc. These macro sets provide cross-assembler capability for a FLEX9 6809 microcomputer host environment for 6800/6801/6802/6808, 6803, 6805, 6502, 8080/8085, and Z-80 target microcomputers. Before presenting the review of this software, I will briefly

present a background to cross-machine software development in order to provide a perspective for the personal computer user.

Cross-machine software development

Often a programmer finds himself in a working environment which requires writing software for more than one type, model, or brand of computer or microprocessor. One solution is to have a separate machine of each type for software development for that particular computer or microprocessor. Another solution is to provide the necessary software tools for software development for all of the varying computers and microprocessors in one uniform environment for the programmer, which means offering these software tools on a single machine.

There are considerations of economy, efficiency, etc. in choosing between these two environments. Certainly for the personal user the latter approach is the most desirable. This kind of software development environment is usually referred to as "cross-machine" support. For example, an assembler for a 6502 microprocessor which actually runs on a 6809 computer is called a "cross-assembler". More than a cross-assembler is needed to provide a software development capability for a "target" machine on a different "host" machine.

In general, a simulator or emulator is provided which allows the object code of the target computer to be executed on the host machine via the simulator. Usually the simulator provides execution trace and debugging facilities to the programmer. The disadvantage is that all testing is not done in a real-time environment which means that the simulator cannot duplicate the actual speed and timing characteristics of the target computer though it can simulate all of the logical functions of the target machine.

Besides the simulator, cross-development aids such as disassemblers, eprom programmers, etc. may be found on a host system computer which provide software development capability for various target machines. It is clear that only a cross-assembler and an editor are needed on a host machine to allow the programmer to write an assembly language program for a target machine, assemble that program, and generate executable object code for the target machine. But it is also clear that a simulator for the target machine is highly desirable since very few programs work properly on the first try.

In this "cross-development" environment the programmer can use the host system's editor and utilities to write his code for a target machine. The host machine is of course the machine he is actually using and the target machine is the computer for which the program is actually intended.

The programmer will then assemble his program using the cross-assembler for the target machine. Naturally, his program is assembly language mneumonics for the target machine and the output of the cross-assembler will be object code which is executable on the target machine. The programmer is faced with the task of physically transporting his object code from his host machine to his target machine, and this may be done via eprom or by some other means. First, the programmer will need to test and debug his program. Since the programmer has a simulator for his target machine he will simply execute his target machine object code via the simulator for the purposes of testing and debugging and then he will transport his code when he is satisfied his program works.

6800/1, 6805, and 6502 CSC cross-assemblers

Computer Systems Consultants, Inc. sells cross-assemblers for other microprocessors as previously mentioned, but this review is confined to the microprocessors for which I have CSC cross-assemblers. Since all of the microprocessor cross-assemblers are constructed similarly, this discussion applies to the other microprocessor cross-assemblers except that I have only tested these particular cross-assemblers.

Now that I have used the name "cross-assembler" freely, let me say that CSC does not sell cross-assemblers per-se. A cross-assembler would truly be a stand-alone assembler running on a host machine which accepts mneumonics and generates object code for a target machine. CSC sells sets of macros for use with the Technical Systems Consultants 6809 macro assembler which together provide cross-assembler functionality. A different macro set is provided for each different target microprocessor. To effectively utilize these cross-assembler macro sets you must have the TSC 6809 macro assembler, though since the macro set is necessarily source code, it is possible to adapt the macro set to another macro assembler. Version 2 of the TSC 6809 macro assembler is needed for use with these macros.

In addition to the sets of macros there is one program called the macro translator which is needed to perform cross-assembly. Briefly, for a FLEX9 environment using the CSC cross-assembler macros you need:

1) TSC 6809 macro assembler, version 2.
2) CSC macro translator program.
3) CSC macro set for the target microprocessor.

Of course, you obviously also need to write an assembly language source code program for the target machine so that you will have a FLEX9 source code file waiting to be converted into object code for the target microprocessor by the above software.

The macro translator is in effect a preprocessor which converts the source code file into a form which is compatible with the TSC macro assembler. For instance, a source code program written for the 8080 microprocessor has a different format from a 6809 source code program, yet the programmer would like to write his 8080 source code in a manner entirely compatible with an 8080 microprocessor environment. With the CSC macro translator the programmer can write his source code in the target machine format. This source file is then processed by the macro translator which generates another source file for input to the TSC 6809 macro assembler. Here are the steps:

1) Prepare a target machine source code assembly language file.
2) Input the target source code file to the macro translator which outputs an intermediate source code file suitable for the TSC 6809 macro assembler.
3) Input the macro translator intermediate source code file to the TSC 6809 macro assembler which outputs a binary object file for execution on the target machine.

The purpose of the macro translator is to allow the programmer to write a target machine assembly language source code program that is entirely

compatible with "native" assemblers for the target machine and yet ultimately utilize the TSC 6809 macro assembler to perform the cross-assembly. The macro translator comes from CSC in source code form and has to be assembled by the user. It is written to be assembled on either a FLEX2 6800 machine or a FLEX9 6809 machine and will adjust itself accordingly to the two environments. It is not necessary for the user to modify the macro translator for either environment. The macro translator functions with all of the macro sets so that there is only one version of the macro translator.

CSC cross-assembler documentation

The printed documentation provided with the cross-assembly software is excellent. The documentation clearly explains the "how-to" aspects of user operation and also explains the functions performed by the macro translator for each of the various microprocessors. The documentation notes all target machine assembly language source code file requirements such as not allowing line numbers in 8080/8085 source code files, and a few other restrictions, none of which conflict with common assembly language environments for the target microprocessors.

Example files are provided on the software disk. These files include a target machine assembly language source code file and the subsequent macro translator source code file for each target microprocessor for which you purchase cross-assembler macro sets. These example files are useful, but disappointing. They are essentially nonsense source code files and you have to place the two files side-by-side and study them to begin to see what the macro translator is doing in processing the original source code into the intermediate source code. It would have been much better to provide a very simple source code program like an ACIA INCH or OUTCH routine, or several of these exceedingly simple routines to demonstrate the macro translator processing. Certainly all of the macro translator functions are demonstrated via the nonsense text files provided, but I think most people prefer to study "real" source code.

CSC cross-assembler operation

To actually use the cross-assembler macros and the macro translator is very simple. Just execute the macro translator (MACXLAT.CMD) and it will prompt for input and output file names. The input file needs a LIB statement at the beginning of the source code for the corresponding macro set. If you are processing code for a 6502 microprocessor then you need to add "LIB MAC6502" to the front of the original source code file. The intermediate source code file output by the macro translator may be immediately assembled using the TSC 6809 macro assembler. The TSC assembler will of course generate a binary object file which is executable binary code for the target microprocessor. That is all there is to it!

I received the macro sets for the 6800/1, 6805, and 6502 micros. I assembled several old 6800 programs and everything went smoothly. Emboldened by this success I turned the software over to a couple of associates at least as butter-fingered as yours truly and asked them to generate some 6805 and 6502 programs. Several programs straight out of some textbooks along with a couple of original programs were produced in short order. The only problems encountered were the normal typos and logic failures associated with these endeavors, but most importantly, the cross-assemblers produced the correct responses in all cases. Certainly this testing was not exhaustive, but the use of these macro sets continues with satisfied users at this end.

Conclusion

Computer Systems Consultants, Inc. of Conyers, Georgia offers an excellent product in these cross-assembler macros. I hope to review additional software from CSC in the near future. In particular, I have received the CSC Super Sleuth suite of programs which perform sophisticated disassembly and editing of binary files and a few other nice functions. I'm sure some of you would like a review of simulators which are generally referred to in this article. Let me hear your comments and suggestions.

By Steven M. Ward
39 Thorndike St.
Arlington, MA 02174

EXTEND A FLEX DIRECTORY

Those of you who use FLEX know that when a disk is initially formatted, track 0 contains the directory sectors starting at sector 5 and ending at the last sector of track 0; sector A. This gives you six sectors of directory entries, and at ten entries per sector, this yields 60 entries.

Well, if you have single sided, single density disk drives, 60 files on one disk will probably do you just fine. Once you start advancing to double sided, double density and even double track drives, the small initial directory size becomes noticable. I have two of these "octo-density" type drives and it is not uncommon for me to have more than 60 files on my 2000 sector capacity diskette. Sure, FLEX will extend the directory automatically after the initial directory sectors are used up, but the additional sectors are taken from the first available free sector, and thus the directory becomes fragmented across the disk. Furthermore, FLEX only extends by one sector at a time, so you really start to notice the extra seeking needed to find these fragmented file entries. The optimum solution is to allocate a large enough directory space when the disk is formatted. The directory will be contiguous and files will be found much faster.

I'm sure a lot of you realize this problem and have been to busy (or too lazy) to write you own utility to solve the problem. Using some kind of repair utility to change the directory links by hand does the job, but is a nuisance and can be disasterous if it is not done carefully.

The EXTEND command was written to solve this very problem. It is designed to be used following a disk format, and will increase the initial six sector directory by 1 to 30 sectors, thus yielding a possible additional 300 directory file entries. This maximum extension value may be changed of course, to suit the individual. Since the program finds the last directory node by chaining through the directory sectors, this command will work with any type of disk (single/double sided, single/double density, 5"/8", etc.), and can be used at other than disk format efficiency of directory searching, use after a disk format. Syntax is described in the program listing, but here are a couple of examples:

```
EXTEND E=20,D=1
```

Extend the directory of the disk in drive 1, by 20 sectors (disk will have 60+200=260 contiguous directory entries initially).

```
EXTEND
```

Extend the directory of the disk in the work drive by 10 sectors.

Scott Fraser
547 Sharron Bay
Winnipeg, Manitoba, CANADA
R2G 0H8

```
                    *
                    * Bureau/Fraser Software Consultants
                    * 1 Pleasant Bay
                    * Winnipeg, Manitoba, Canada
                    * R2K 0C9
                    * July, 1982
                    *
                    * This EXTEND command takes a newly formatted disk
                    * and adds several more sectors to the directory.
                    * The maximum allowable amount to extend by is 30
                    * sectors (yields an additional 300 directory entries).
                    *
                    * It is called as:  EXTEND [D=drv#][,E=#secs]
                    *
                    *   Where:
                    *           D=   specifies the drive to extend
                    *           E=   specifies the # of sectors to extend by
                    *
                    * If no parameters are given, then the diskette
                    * on the work drive will be extended by
                    * 10 sectors (good for 60+100=160 file entries).
                    *
                    * Note also that since the end of the directory is
                    * found by following the linked chain, and that
                    * the extension is carried out by following the linked
                    * nodes, this command will work with any type of
                    * diskette (single/double sided, single/double density,
                    * 5"/8", etc).
                    *

0000                        ORG   $0000
0000                DIRFCB  RMB   FCBLEN    directory FCB
0140                SIRFCB  RMB   FCBLEN    System Information Record FCB
C100                        ORG   $CA
C100 20  09         START   BRA   START1

C102 02                     FCB   2         version 2

             000A   DEFSEC  EQU   10        default # sectors to extend by
             0001   MAXDRV  EQU   1         maximum drive number
             001E   MAXXTD  EQU   30        max # of sectors to extend by

C103                DRV     RMB   1         holds drive number
C104 00             DPSEC   FCB   0         for 16 bit subtract
C105                SECTRS  RMB   1         holds sector extension amt
C106                TRKSEC  RMB   2         holds trk/sec value
C108                CTR     RMB   1         counter
C109                DIREND  RMB   2         holds trk/sec of 1st dir node

             C10B   START1  EQU   *
                    *
                    * First setup the default drive number and the
                    * default number of sectors to extend the directory
                    * by.
                    *
C108 B6  CC0C               LDA   #MAXDRV   default is work drive
C10E B7  C103               STA   DRV       save it
C111 B6  0A                 LDA   #DEFSEC   default # of sectors
C113 B7  C105               STA   SECTRS    save it
                    *
                    * Now parse for the input parameters
                    *
C116 BD  C204               JSR   CHKPRM    set parms
C119 1025 00C9              LBCS  ERR01     bad parm -> error

C11D B6  C103               LDA   DRV       get drive # to print
C120 8A  30                 ORA   #'0       convert to ascii
C122 B7  C332               STA   OUTDRV
C125 8E  C314               LDX   #RDYMSG   print msg before starting
C128 BD  CD1E               JSR   PSTRNG    print msg
C12B BD  CD09               JSR   INCH      and get response
C12E B4  5F                 ANDA  #$5F      convert to upper case
C130 81  59                 CMPA  #'Y       if yes then continue on
C132 1026 00AB              LBNE  EX005     if anything else then exit
                    *
                    * Set drive in FCB's
                    *
C136 8E  0000               LDX   #DIRFCB   point to directory FCB
C139 B6  C103               LDA   DRV       get drive number
C13C A7  03                 STA   FCBDN,X   save in directory FCB
C13E 8E  0140               LDX   #SIRFCB   point to SIR FCB
C141 A7  03                 STA   FCBDN,X   save in SIR FCB

C143 CC  0003               LDD   #SIRTS    point to SIR
C146 BD  C252               JSR   READ1     get SIR
C149 1026 009E              LBNE  ERR02     report error
                    *
                    * The system information record has been read.
                    * Extend the directory by the number of sectors
                    * specified.  To do this, the last sector of the
                    * directory must be pointed to the first sector in the
                    * free chain.  The link of the last extended sector
                    * in the directory must be zeroed.  The SIR must
                    * then have its free chain pointer updated, as well
                    * as its "number of free sectors" value.
                    *
C14D 8E  0000               LDX   #DIRFCB   point to directory FCB
C150 CC  0005               LDD   #DIRTS    point to beginning of dir
             C153   EX010   EQU   *
C153 FD  C109               STD   DIREND    save link
C156 BD  C252               JSR   READ1     get sector
C159 1026 008E              LBNE  ERR02     report error
C15D 8C  88 40              CMPD  SBLINK,X  get link
C160 26  F1                 BNE   EX010     if <>0 then continue search
                    *
                    * Found the end of the directory chain.
                    * Update this to point to the first sector of the
                    * free chain
                    *
C162 108E 0180              LDY   #SIRFCB+FCBSB point to SIR's sec buf
C166 EC  A8 1D              LDD   SIRFSB,Y  get first free sector
C169 ED  88 40              STD   SBLINK,X  and save in directory node
                    *
                    * Write out updated dir sector
                    *
C16C 8E  0000               LDX   #DIRFCB
C16F FC  C109               LDD   DIREND
C172 BD  C25C               JSR   WRITE1    write back this entry
C175 26  7C                 BNE   ERR03     report error
                    *
                    * Now chain through the free space chain "sectrs"
                    * times.  Once at end, zero its link
                    *
C177 B6  C105               LDA   SECTRS    # sectors to extend by
C17A B7  C108               STA   CTR       save away
C17D EC  A8 1D              LDD   SIRFSB,Y  get ptr to first free link
C180 8E  0000               LDX   #DIRFCB   point to an FCB
             C183   EX011   EQU   *
C183 FD  C106               STD   TRKSEC    save this link
C186 BD  C252               JSR   READ1     read a sector
C189 26  60                 BNE   ERR02     report error
                    *
                    * Before going on to the next node,
                    * zero out this one's data area
                    * and write back on disk (ensures
                    * a "clean" directory)
                    *
C18B 86  FE                 LDA   #$254     # bytes of data to zero
C18D 30  88 42              LEAX  SBRS1,X   point to area
             C190   EX013   EQU   *
C190 6F  80                 CLR   0,X+      zero a byte
C192 4A                     DECA
C193 26  FB                 BNE   EX013     continue until done
C195 8E  0000               LDX   #DIRFCB   point back to FCB
C198 7A  C108               DEC   CTR       extended enough?
C19B 27  0F                 BEQ   EX012     yes, then quit search
C19D FC  C106               LDD   TRKSEC    recall its trk/sec
C1A0 BD  C25C               JSR   WRITE1    write back to disk
C1A3 1026 0008              LBNE  WRERR     on error quit
C1A7 EC  88 40              LDD   SBLINK,X  nop, get next link
C1AA 26  D7                 BNE   EX011     keep going if not zero
             C1AC   EX012   EQU   *
C1AC 8C  88 40              CMPD  SBLINK,X  start of free space
```

```
C1AF 27 4A          BEQ   ERR04     if no room left then error!
C1B1 ED A8 10       STD   SIRFSB,Y  save in SIR
C1B4 EC A8 21       LDD   SIRFSS,Y  get size of free chain
C1B7 B3 C104        SUBD  OPSEC     sub off extension size
C1BA ED A8 21       STD   SIRFSS,Y  save back to SIR
                    *
                    * Write out the last directory node
                    * (in DIRFCB) but zero its linkage field.
                    *
C1BD CC 0000        LDD   #$0000    zero link field
C1C0 ED 88 40       STD   SBLINK,X
                    *
                    * Write updated sector
                    *
C1C3 FC C106        LDD   TRKSEC    get trk/sec to write
C1C6 BD C25C        JSR   WRITE1
C1C9 26 28          BNE   ERR03     report error
                    *
                    * Write out updated SIR
                    *
C1CB 8E 0140        LDX   #SIRFCB   point to SIR FCB
C1CE CC 0003        LDD   #SIRTS    point to SIR
C1D1 BD C25C        JSR   WRITE1    write it out
C1D4 26 1D          BNE   ERR03     report error
                    *
                    * All done
                    *
C1D6 8E C20F        LDX   #DONE     done msg
C1D9 BD CD1E        JSR   PSTRNG    print it
C1DC 8E C104        LDX   #OPSEC    print out # secs extended
C1DF 5F             CLRB            suppress leading zeroes
C1E0 BD CD39        JSR   OUTDEC    print number
            C1E3 EX005  EQU   *
C1E3 7E CD03        JMP   WARMS     return to flex
                    *
                    * Error routines
                    *
            C1E6 ERR01  EQU   *
C1E6 8E C266        LDX   #BADPRM   get bad parameter msg
C1E9 20 13          BRA   ERR       print and return

            C1EB ERR02  EQU   *
C1EB BD CD3F        JSR   RPTERR    report error first
C1EE 8E C274        LDX   #RDERR    get read error msg
C1F1 20 0B          BRA   ERR       print and return

            C1F3 ERR03  EQU   *
C1F3 BD CD3F        JSR   RPTERR    report error first
C1F6 8E C27F        LDX   #WRERR    get write error msg
C1F9 20 03          BRA   ERR       print msg and return

            C1FB ERR04  EQU   *
C1FB 8E C288        LDX   #TOOBIG   get dir too big msg
            C1FE ERR    EQU   *
C1FE BD CD1E        JSR   PSTRNG    print msg
C201 7E CD03        JMP   WARMS     return to FLEX
                    *
                    * Name      - CHKPRM
                    * Function - This routine extracts the optional
                    *            parameters from the line buffer for
                    *            the EXTEND cmd, and saves the values
                    *            in their appropriate places
                    *
                    * On Exit -> if a bad parm was found, then carry is
                    *            set, otherwise clear
                    *
                    * All registers are preserved
                    *
            C204 CHKPRM EQU   *
C204 34 36          PSHS  X,Y,D     save regs
C206 86 03          LDA   #3        Number of possible parms
C208 B7 C108        STA   CTR       save in temp
            C20B CHK003 EQU   *
C20B BD CD27        JSR   NXTCH     get a char
C20E 81 0D          CMPA  #CR       a cr?
C210 27 37          BEQ   CHKOK     yes, then done
C212 81 44          CMPA  #'D       is this a drive spec?
C214 26 16          BNE   CHK001    nop, try next parm
C216 BD CD27        JSR   NXTCH     skip "=" sign
C219 BD CD42        JSR   GETHEX    get drive number
C21C 5D             TSTB            check if number there
C21D 27 25          BEQ   CHK002    nop
C21F 1F 10          TFR   X,D       get drive #
C221 1083 0001      CMPD  #MAXDRV   valid range?
C225 22 27          BHI   CHKBAD    Nop
C227 F7 C103        STB   DRV       save drive number
C22A 20 18          BRA   CHK002    get next parm
            C22C CHK001 EQU   *
C22C 81 45          CMPA  #'E       is this the extend parm?
C22E 26 1E          BNE   CHKBAD    nop, then unknown option
C230 BD CD27        JSR   NXTCH     skip "=" sign
C233 BD CD48        JSR   INDEC     get # sectors to extend
C236 5D             TSTB            check for # there
C237 27 08          BEQ   CHK002    nop, find next parm
C239 1F 10          TFR   X,D       get # sectors
C23B 1083 001E      CMPD  #MAXITD   valid range?
C23F 22 00          BHI   CHKBAD    nop
C241 F7 C105        STB   SECTRS    save value
            C244 CHK002 EQU   *
C244 7A C108        DEC   CTR
C247 26 C2          BNE   CHK003    continue parm check
            C249 CHKOK  EQU   *
C249 1C FE          CLC             set good RC
C24B 35 36          PULS  X,Y,D     restore registers
C24D 39             RTS             return

            C24E CHKBAD EQU   *
C24E 1A 01          SEC             set bad RC
C250 20 F9          BRA   CHKRET    return
                    *
                    * Name    - READ1
                    * Function- This routine reads in the track/sector
                    *           of the disk in the drive specified by an
                    *           FCB pointed to by the X ixr.  Reg D
                    *           specifies the trk/sec to read in
                    *
                    * On Exit -> the carry is set if a read error,
                    *            otherwise clear
                    *
                    * All registers except A are preserved
                    *
            C252 READ1  EQU   *
C252 ED 88 1E       STD   FCBCP,X   set trk/sec
C255 86 09          LDA   #XRSS     set function code
C257 A7 84          STA   FCBFC,X   set code in FCB
C259 7E D406        JMP   FMSCAL    read and return
                    *
                    * Name      - WRITE1
                    * Function - This routine writes a trk/sec given in
                    *            reg D to the drive described by an FCB
                    *            pointed to by the X ixr.
                    *
                    * On Exit -> the carry is set on a write error,
                    *            otherwise clear
                    *
                    * All registers except A are preserved
                    *
            C25C WRITE1 EQU   *
C25C ED 88 1E       STD   FCBCP,X   set trk/sec
C25F 86 0A          LDA   #WSS      set code in FCB
C261 A7 84          STA   FCBFC,X
C263 7E D406        JMP   FMSCAL    write and return
                    *
                    * Data area for strings
                    *
C266 42 61 64 20    BADPRM FCC    /Bad Parameter/
C273 04             FCB   EOT
C274 52 65 61 64    RDERR FCC     /Read Error/
C27E 04             FCB   EOT
C27F 57 72 69 74    WRERR FCC     /Write Error/
C289 04             FCB   EOT
C28B 44 69 72 65    TOOBIG FCC    /Directory takes up all of disk/
C2A9 0D0A           FCB   CRLF
C2AB 50 6C 65 61    FCC   /Please reformat disk and try again!/
```

```
C20E 04                     FCB   EOT
C2CF 0D0A          DONE     FDB   CRLF
C2D1 45 78 74 65            FCC   /Extension Completed Successfully/
C2F1 0D0A                   FDB   CRLF
C2F3 4E 75 6D 62            FCC   /Number of Sectors Extended by = /
C313 04                     FCB   EOT
C314 45 78 74 65   RDYMSG    FCC   /Extending directory on drive #/
C332               OUTDRV   RMB   1
C338 2C 20 61 72            FCC   ", are you ready (Y/N)? "
C34A 04                     FCB   EOT
                            END   START

0 ERROR(S) DETECTED

SYMBOL TABLE:

ABOBX  CD36  ASREAD 0001  ASWRIT 0002  BAC    0008  BADPRM C266
BAK    0005  BAS    0003  BELL   0007  BIN    0000  BS     CC00
BSE    CC07  BUFPNT CC14  CHK001 C22C  CHK002 C244  CHK003 C208
CHKBAD C24E  CHKOK  C249  CHKPRM C204  CHKRET C24B  CLASS  CD21
CLN    0C1A  CLOCK  F700  CMD    0002  CMDFLG CC28  COC    CC29
COLOS  CD00  CR     000D  CRLF   000A  CTR    C108  CURC   CC1B
DAT    0007  DDRV   DE00  DEFSEC 000A  DEL    CC01  DEPTH  CC03
DIR    0009  DIREND C109  DIRFCB 000D  DIRTS  0005  DOBRKD CD4B
DONE   C20F  DOS    CC00  DPSEC  C104  DRV    C103  EJECT  CC08
ENV    CC2D  EOL    CC02  EOT    0004  ERR    C1FE  ERR01  C1E6
ERR02  C1EB  ERR03  C1F3  ERR04  C1FB  ESC    CC0A  ESORR  CC16
EX005  C1E3  EX010  C153  EX011  C183  EX012  C1AC  EX013  C190
FACP   0010  FADP   0040  FARP   0020  FAMP   0080  FCBAS  0002
FCBASE 0409  FCBCDA 002F  FCBCP  001E  FCBDRN 0020  FCBCUR D408
FCBDI  0022  FCBDN  0003  FCBEDA 0013  FCBESB 0001  FCBFA  000F
FCBFC  0000  FCBFCD 0019  FCBFDD 0032  FCBFS  0015  FCBFSM 0017
FCBLEN 0140  FCBLP  001C  FCBNAM 0004  FCBNMB 0024  FCBR1  0023
FCBRSI 0010  FCBRS2 001B  FCBSB  0040  FCBSCF 003B  FCBSCR 0035
FCBSDA 0011  FCBVER D435  FCDDAY 001A  FCDMTH 0019  FCDYR  001B
FIA    CC26  FIEF   CC2F  FLEX   CD00  FMS    D400  FMSCAL D406
FMSYS  D403  FMSERR CC20  FMSINT D400  FOA    CC24  FSMRAN 0002
FSMSEQ 0000  GETCHR CD15  GETFIL CD2D  GETHEX CD42  INBUF  CD1B
INCH   CD09  INCH2  CD0C  INDEC  CD48  IOFLG  CC21  ISWTCH CC23
LAD    CC1B  LF     000A  LNEBUF C080  LOAD   CD30  LSTRM  CC11
MAP    CC00  MAXDRV 0001  MAXXTO 001E  MEMEND CC2B  NULL   CC05
NXTCH  CD27  OSWTCH CC22  OUT    000B  OUTADR CD45  OUTCH  CD0F
OUTCH2 CD12  OUTDEC CD39  OUTDRV C332  OUTHEX CD3C  PAU    CC09
PCRLF  CD24  POUT   CCE4  PRCHK  CCD3  PREVC  CC19  PRINIT CCC0
PRT    000A  PSTRNG CD1E  PUTCHR CD1B  RDERR  C274  RDYMSG C314
READ1  C252  RENTER C006  RPTERR C08F  RSTRIO C02A  SBDATA 0044
SBLINK 0040  SBRS1  0042  SDFNSC 00FF  SDFSC  0000  SCR    0006
SECTRS C105  SETEXT CD33  SFA    C980  SIRCRE 0023  SIRDAY 0024
SIRFCB 0140  SIRFSB 001D  SIRFSE 001F  SIRFSS 0021  SIRLEN 0028
SIRMTH 0023  SIRMTS 0026  SIRNAM 0010  SIRTS  0003  SIRVOL 001B
SIRYR  0025  SP     0020  SPS    C700  START  C100  START1 C108
STAT   CD4E  STKA   D000  SYDR   CC0E  SYDRV  CC0B  SYS    0004
SYSCDN CC4E  SYSCR1 CC0D  SYSDR2 CC2A  SYSCR3 CC30  SYSCR4 CCF8
SYSFCB C840  TAB    CC06  TOOBIG C288  TRADDR CC1E  TRFLG  CC1D
TRKSEC C106  TXT    0001  UCA    C100  UCTA   CC12  URAM   0000
WARMS  CD03  WIDTH  CC04  WKDRV  CC0C  WRERR  C27F  WRITE1 C25C
XBOR   0016  XCLOSE 0004  XDELET 000C  XFND   0014  XGIR   0007
XGRB   0011  XNSS   000F  XODIR  0006  XOREAD 0001  XOSIR  0010
XOUPDT 0003  XOWRIT 0002  XPIR   0008  XPOSN  0015  XPRB   0012
XRENAM 0000  XRES1  000B  XRES2  000E  XRES3  0013  XREWND 0005
XRSS   0009  XRWNB  0000  XWSS   000A
```

READING HARD SECTOR DISKS

Sept. 2, 1984

P.O. Box 6492 Stn. 'J'
Ottawa, Ontario
K2A 3Y6

Dear Sir,

The circuit enclosed was designed a year ago when I was faced with having to throw away hundreds of hard-sectored diskettes in favor of soft-sectored types. Since that was an expensive option, the circuit allowed me to continue to use and mix diskettes freely and I saved a lot of money. Since your magazine may have readers with a similar situation, I thought that I had better send you the circuit than let it gather dust here.

The circuit connects between the actual disk data decoding circuits and a typical Western Digital 1771, 1791, and even 2791 chip. Only one revolution is required to determine whether a disk is hard or soft sectored and the effect is identical to covering up the sector holes with labels. I have only used this circuit on 5" drives but there is sufficient room for adjustment to make it run with 8" diskettes.

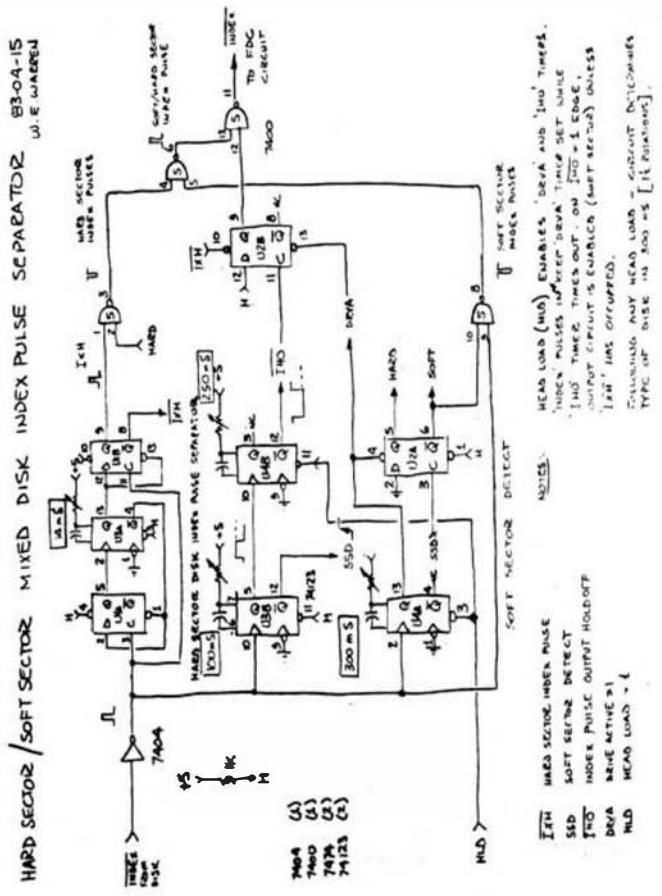
Your magazine, although not as thick as those for the other processors still has maintained a steady flow of useful information and although I inadvertently let my subscription lapse, I will be renewing shortly.

Thank you for your attention, and I remain...

Yours truly,

William Warren

William E. Warren



US ALL !

SURVEYS AND YOU

We have recently completed another survey for a large national computer company. Most of the replies were about as I expected. This survey contacted approximately 3,000 68XX(X) users. I want to thank you again for your cooperation, without your assistance we would not be able to keep as accurate profiles as we do.

Accuracy; well, ABC this week (Sept 13 '84) announced a 'nation wide, comprehensive' voter survey - they also surveyed 3,000 - nation wide! I would assume that our percentages could also be termed comprehensive!

Three thousand (plus/minus 50) of you gives it about a +/- factor of 2%, according to an ad agency we work with. From the continued interest I receive from computer manufacturers, who are not selling to our group presently, I can't help but feel that we have a long way to go, despite what the doom-sayers are muttering. Sometimes I get the feeling that some who presently sell here, but are singing the blues about our 'sliding down the tube' are just paving a smooth way for them to abandon the group who made them, namely us, we who furnished the capital through our purchases and hung in there loyally. Oh, well.

I believe that we have a long way to go, with or without those who feel we are all used up. But it isn't us who are all used up, I firmly believe, no, I know what the reason is....we are still here kicking, there just has not been any significant new products recently. Especially in software. New products - state of the art products are what makes things happen.

Fact, our (68 Micro Journal) subscriptions are not declining, we are still showing gains. Not as rapid as last year, or the year before, but climbing. Meaning, the market is still there, most of us have too much invested to chuck it all, and lay out thousands for new computers, much less having to learn new languages, operating systems and of course, the heavy expense of buying all that new software, and learning how to use it. What we need, and are not getting is new products, and improved versions of some of our old ones (many can certainly stand some sprucing up and fixing), both soft/hardware wise. However, I guess we need to let some of our suppliers know.

Most of the information garnered by our most recent surveys is confidential. The sponsor(s) paid good money to get the survey done and expects the results to be for their benefit exclusively. That they paid for and that they will get. However, I want to do a small, informal survey just for our advertisers and suppliers. I want them to hear from you, and let you tell them what you want and are willing to pay for! I will furnish the results to any current or past advertiser for the asking. I will even pay postage and all survey cost. But I want them to KNOW that it is YOU speaking. Maybe, just maybe some of them will hear. I hope so, before it too late for some of them, and more important, us!

Already I have a pathetic letter, from a past advertiser, who stopped supporting his excellent S50 bus products, and spent all his available capital and cash-flow revenues trying to put his product on two 'other side' computers. Today he is bankrupt, and not only is he a loser, but so are we. We probably won't get any more 'fixes' to his products that we bought and still use and even worse we will never get a chance to use some of the other 'good' projects he had in mind before the switch. All in all, we are the worst hit losers, there is just one of him but thousands of us!

I am really concerned about this sort of thing. I only wish that most would investigate what the real cost are before they leap. It is a lot more expensive to do it 'over there' than here on the S50 bus. Many are excellent programmers and hardware designers, but few are also good business types. Sad but true, and while we (S50 bus) users have tolerated rips here and holes there, the 'other side' will not. Think about it! So, if you care Please fill in the short survey below and send it back to me within 30 days - Thanks.

DMW

INFORMAL SURVEY

1. What make computer(s) do you use _____

2. What disk system (FLEX™,OS-9™,Star-Dos™) _____

Other disk system _____

3. What size disk (8-5 inch, hard-disk) _____

4. What major software _____

_____

_____

5. What software would you buy that is not now available

_____

_____

_____

6. What hardware would you buy that is not now available

_____

_____

_____

7. Are you going to invest any large sum on computers from the 'other side' - Yes/No

_____

8. If 'Yes' to #7 - what - when - how much _____

_____

_____

_____

_____

10. What do you want to say to our advertisers about DISK SYSTEMS, HARDWARE, SOFTWARE, DOCUMETATION, SUPPORT, NEW PRODUCTS, ETC. - (please use additional sheets of paper as necessary) This is the important question

_____

_____

_____

_____

_____

11. What do you use your 68XX(X) computer for - what application - primarily

_____

_____

_____

_____

12. What suggestions do you have for us (68 Micro Journal)

_____

_____

_____

DOCUMENTATION , THE NECESSARY EVIL

DOCUMENTATION, THE NECESSART EVIL

Wilbur N. Killebrew, Jr.
14735 Kollywood Lane
Houston, Texas 77079
(713) 558-1877

Somewhere there is a computerist -- large system or small -- who actually enjoys documenting his system, does it well and does not spend an undue amount of time at it. Most of us do what must be done at the moment and then spend a lot of time "retraining" ourselves when the need arises. Murphy, the optimist he is, usually obliges through the operation of his laws, supplying the appropriate emergency at the proper time, the affair becomes hectic and when the dust and feathers have settled, viola! The state of affairs is unchanged -- the documentation is still inadequate.

Given the quality of documentation supplied by most manufacturers, even the biggest and "best," it would be asking a lot of a microcomputer owner/user to do a thoroughly adequate job. The worst part of the problem seems to be identifying what is really needed. Unless we include making provision for proper safe-keeping.

If you have a totally prepackaged system provided by a major house, there is some small chance that you need to do little or nothing. Most of us who own SS-50 based machines are in the same position as owners and users of minicomputer systems have always been. You do it yourself or there isn't any. Lotsa-luck!

Not everyone learns or retains information the same way, but for me it has always been useful to write-things-down. A refinery operations superintendent once explained the phenomenon in an interesting way. The subject at the time was "the need for plant operators to keep manual log sheets." It goes like this: When a person writes something down, the information flows up the writing instrument, through the arm and then into the brain, where it is stored and remembered. A little unorthodox, perhaps, but most of us do tend to remember what we have written down. That is, if any effort was made at all to give adequate concentration to what was being written.

Perhaps you do not need (or want) to remember so much about your system. That's fine, but some time there will be a need for it. Even if you do not remember all the details, concise, well organized notes will make the recall process operate much faster than having to pore over basic vendor documentation.

One way to ease this task is through users groups and magazines such as "68" Micro Journal; by many users sharing bits and pieces of information of mutual interest. Unless it is gathered into some sort of repository, however, it can be a real bear to find a neat tidbit that is needed. Remembered but not available.

Quick-Reference Sheets

It seems that almost every time I work on a new project of any complexity, I end up making one or more reference sheets for myself. That's what most user documentation is for me, quick reference sheets. All the essential facts are tabulated and organized and annotated for quick reference.

In the interest of furthering similar activities, I am publishing some of the documentation I have prepared for my own use. Three of the Tables are based on the TSC FLEX manuals or SBUG-E documentation. The only unique feature is its quick-reference format, much like the instruction set cards which computer manufacturers supply. There is still the need for the manuals, but I find that some 80% of the information I need is in these quick reference sheets. If I could make larger sheets and reduce them to 8.5 x 11, more could be put on them. I made two-sided copies of the original versions of the FLEX reference tiles, with the FCB and FMS tables reduced to 72% and put together on the back. Since the original is pica (10 characters to the inch), the reduced versions are still quite easily readable. There is an item which has not been added to the FMS function codes table that I ran across not too long ago. TSC shows FMS function 17 ($08) in their FLEX manual as "reserved for future system use." I haven't investigated, but the one instance of its use I have run across seems to indicate that it is "update catalog entry," or something very close to that.

The quick reference chart for RS-232C signals has its rows and columns transposed from the layout used in the original. Although it is not quite as easy to read, this arrangement allows unlimited growth, while the other is less flexible. If there is any demand to have these published, I think I would prefer to go back to the vertical arrangement, as it it a lot easier to read.

There are a lot more SS-30 boards available than are shown in the tables here. Is it worth our while to publish more nearly complete listings of them?

Serial Communications Interfaces

Serial interfaces always seem to be a problem, so that's the reason I have made reference sheets for serial I/O. This one no doubt gives me away as do-it-yourselfer. There isn't a single reference to SWTP, SSB or Gimix in the lot. I have built all the boards in my system, but one. The problem with serial I/O is total lack of agreement as to what should constitute a standard set of signals for a terminal interface, or even whether the connector should be male or female in any given situation. So with each new hookup, it is necessary to get out the reference materials to determine how to make up a cable.

A lot of people mistakenly believe that EIA standard RS-232C specifies a lot more than it does. It specifies the signals that may be used, but does not necessarily require any specific combination. It also specifies the sense (active high or low) of each and gives the maximum and minimum voltages which may constitute a mark (one) or a space (zero). Until recently, nearly all vendors have used the DB25 connectors for RS-232C, but the specification itself does not require any specific connector. To shave a few cents off the cost of their micros, Radio Shack uses a round connector and Commodore a DE9. I believe ATARI uses a DA15. By the way, many refer to all D-subminiature connectors as DB's. Only the 25 pin connector is correctly designated DB, as in DB25. The others are: DE9, DA15, DC37 and DD50. All but the DD50 have two rows of pins, the upper having one more pin than the lower. The DD50 has three rows of pins, the middle one having only 16 pins.

In the process of wrapping this up and making the drawings, I looked over some of my stock of D-subminiature connectors. As it to make me the liar, one of them, an M-D plug, has the legend "DB25S" clearly molded between the very obviously male pins! The TRW Cinch catalog listing labels this device "plug," with a part number of DB25P. But says that the P stands for "pin." Amphenol and AMP both just use long numbers.

Some Manufacturer's Numbers for D-Subminiature 25 Pin Connectors

Manufacturer	Cable Termination	Pin socket housing (receptacle)	Pin housing (plug)
AMP Spl. Ind.	Shell, no contacts	205207-1	205208-1
AMP Spl. Ind.	Ribbon cable IDC	206770-1	206771-1
Amphenol	Removable solder p	17-10250	17-20250
Amphenol	Fixed solder pot	17-80250	17-90250
M-D	Fixed solder pot		DB-25S
ITT Cannon	Fixed solder pot	DB-25S	DB-25P
T&B Ansley	Ribbon cable IDC	609-25S	609-25P
TRW Cinch	Fixed solder pot	DB-25S	DB-25P
Winchester	Ribbon cable IDC	49-1125S	49-1125P

Parallel Communications Interface

If you have never hooked up a "Centronics" parallel interface, how do you know what connector(s) to buy? The folks in the big cities don't know the true joy of do-it-yourself-ing. They just go down to the local CompuShack Shack, look things over and buy the best deal. Nearly all my system was put together while I was in Eastern Kentucky. Practically everything was obtained by UPS, sight unseen. The choice is to pay $40 for a cable that MIGHT work, or to find out what is needed. I side-stepped the issue and bought an Okidata B3A, with both serial and parallel interfaces "at no extra cost." Since I had no trouble hooking my H-19A terminal up with a serial interface, that is the first one I tried. It worked so well I have never tried the parallel interface. But I did dig out the connector information for the so-called "Centronics standard" parallel interface. It turns out that Centronics uses a number of different connectors, not just one as implied by most of the marketing hype. Most printers advertised as having a "Centronics" parallel interface do have the same type, however. It is a 36-contact female micro-ribbon connector, of the panel-to-cable type. Part numbers for some manufacturers are listed in the table. Also shown are part numbers for the corresponding male plug which is needed for the cable. I had considered using these connectors on my computer back panel, but the micro-ribbon connectors tend to be expensive. Advertised prices have fallen to less then $8.00 recently, making them more competetive with the familiar D-subminiature connectors, so I may use micro-ribbons anyway.

Communications Interface Documentation

I didn't intend to make connector drawings, but since a picture is worth a good many words, I have relented. The reference I was going to use shows one incorrectly and there is no designation as to whether the drawings are of male or female connectors, nor whether the views are frontal or rear. In the final analysis it doesn't make that much difference as long as the correct contact assignments are available. The connectors have the contact numbers molded on them. If you can read them. Sketches of both the micro-ribbon and D-subminiature connectors are included.

The Black Box Catalog company, P O Box 12800, Pittsburg, PA 15241 issues a catalog which contains nice pinout tables for some of the most commonly encountered "standard" connectors. It has tables for RS-232C and RS-449, also showing the CCITT V.24 designations equivalent to RS-232C. It also has drawings of RS-449, RS-232C/V.24, CCITT V.35 and "Centronics" parallel (36 contact) connectors. Unfortunately, there is no designation as to whether these are male or female connectors. There IS a difference, you know, besides the obvious one of "sex." They are mirror images of each other. By the way, do you know that ATARI's "Centronics" connector has 50 contacts? How many other "standard Centronics parallel" connectors are there? I haven't tried to make a survey of the different connectors Centronics uses beyond checking that Centronics actually uses both the 36- and 50-contact units.

The Black Box drawings are for the frontal view of DB25P and DC37P connectors (or the rear view of DB25S and DC37S if you prefer to think in terms of looking at female behinds). The drawing of the Centronics connector appears to be of a plug (male). If

it is, not only are the contects numbered in the wrong direction, but contect 1 is shown on the narrow side of the D-shaped shell, instead of on the wide side. No matter how you look at it, it is wrong. I don't know about the V.35 connector; I have never used one, nor seen any other documentation on it. It has 34 pins, arranged in four rows, alternately 9 and 8 to the row.

In addition to the connector and signal descriptions, Black Box Catalog's catalog has a two page communications dictionary.

In the interest of completeness and accuracy, the following definitions are included. Perhaps some beginners will find them useful:

Socket = Female = suffix S, has pin receptacles
Plug = Male = suffix P, has pins

This breaks down in the case of the micro-ribbon connectors since the contects of the male connector are not pins. Both the male and female connectors are fitted with small ribbon springs which nestle against each other when the connectors are mated. On the male connector they are on the outside perimeter of a plug, while the female contects are on the inside perimeter of a cavity. The socket cavity is rectangular, but the shells of both the male and female connectors are D-shaped similar to the D-subminiature connectors, ensuring correct mating.

This plug and socket thing, while quite elementary, can be a little confusing. The D-subminiature socket unit is plug-shaped and fits within the shroud of the plug unit. The perspective is, however, the pins and the mating sockets, rather than the shells in which they are mounted.

FLEX Error Messages

#	Message
1	Illegal FMS Function Code
2	Requested File is in Use
3	File already Exists
4	File could not be found
5	System Directory Error
6	System Directory is Full
7	All Avail Disk Space Used
8	Read Past End of File
9	Disk File Read Error
10	Disk File Write Error
11	File or Disk Write Prot'd
12	File Prot'd - Not Deleted
13	Illegal File Contrl Block
14	Illegal Disk Access
15	Illegal Drive Number
16	Drives Not Ready
17	File Protected, no Access
18	System File Status Error
19	FCB Data Index Error
20	FMS not Active - Reboot
21	Illegal File Specific'n
22	System File Close Error
23	Sector Map Overflow *
24	Non-Existent Record No.
25	Record No. Match Error
26	Command Syntax Error
27	Cmd Not Allowed While Printing
28	Wrong Hardware Configuration

FMS Functions

# hex	Function
0 $00	Read/Write Next Byte/Char
1 01	Open for Read
2 $02	Open for Write
3 $03	Open for Update
4 $04	Close File
5 $05	Rewind File
6 $06	Open Directory
7 $07	Get Information Record
8 $08	Put Information Record
9 $09	Read Single Sector
10 $0A	Write Single Sector
11 $0B	-
12 $0C	Delete File
13 $0D	Rename File
14 $0E	-
15 $0F	Next Sequential Sector
16 $10	Open System Info Record
17 $11	Get Rand Byte from Sect
18 $12	Put Random Byte in Sect
19 $13	-
20 $14	Find Next Drive
21 $15	Position to Record N *
22 $16	Back up One Record *

Note: * indicates applies only to random files.

FLEX 9.0 File Control Block (FCB) Map

Byte Number	Description	Remarks
0	$00 FMS Function Code	Less than 22 ($16)
1	$01 FMS Error Code	0 = no error
2	$02 Activity R/W Code	1 = Read, 2 = Write
3	$03 Drive Number	0 < Drive Number < 4
4-11	$04-0B File Name	
12-14	$0C-0E Extension	
15	$0F Attributes (Protection)	b7=W, b6=D, b5=R, b4=C
16	$10 -	
17-18	$11-12 Disk File Start Address	TTSS
19-20	$13-14 Disk File End Address	TTSS
21-22	$15-16 File Size (Sectors)	Incl. sector map if random
23	$17 File Sector Map Code	0 = sequential, 2 = random
24	$18 -	
25-27	$19-1B File Creation Date	MMDDYY
28-29	$1C-1D Next FCB Pointer Addr	Points to Pointer not base
30-31	$1E-1F Current Disk Address	TTSS
32-33	$20-21 Current Record Number	Sector sequence no in file
34	$22 Data Index (DI)	Next sequential byte
35	$23 Random Index	Byte no. for random access
36-46	$24-2E Name Work Buffer	FMS use only
47-49	$2F-31 Curr Directory Address	TTSS and Data Index in CAT
50-52	$32-34 1st Deleted CAT Pointer	TTSS and Data Index in CAT
53-63	$35-3F Scratch bytes	Hex NAME,EXT for Renaming
59	$3B Space Compression Flag	0 = compress, $FF = no
64-65	$40-41 Logical Record Number	Bytes 142 in sector DI=0,1
66	$42 Track Number (TT)	3rd byte in sector, DI = 2
67	43 Sector Number (SS)	4th byte in sector, DI = 3

Quick-Reference Card for TSC FLEX 9.0

DOS Working Storage

$C080-FF Line Buffer

File Management System Entries

$CD00 COLDS DOS Cold Start
$CD03 WARMS DOS Warm Start
$CD06 RENTER DOS Re-entry
$CD09 INCH Input Character
$CD0C INCH2 Input Character
$CD0F OUTCH Output Character
$CD12 OUTCH2 Output Character
$CD15 GETCHR Get Character
$CD18 PUTCHR Put Character
$CD1B INBUFF Inpt to Line Buff
$CD1E PSTRNG Print Char String
$CD21 CLASS Classify Char
$CD24 PCRLF Print CR & LF
$CD27 NXTCH Get Next Buff Char
$CD2A RSTRIO Restore I/O Vctrs
$CD2D GETFIL Get Filenam to Buf
$CD30 LOAD Load Binary File
CD33 SETEXT Set Default Exten
$CD36 ADDBA Add (B) to (X)
$CD39 OUTDEC Output Decimal No
$CD3C OUTHEX Output Hex Number
$CD3F RPTERR Report File Error
CD42 GETHEX Get Hex Number
$CD45 OUTADR Output Hex Number
$CD48 INDEC Input Decimal No.
CD4B DOSCMND Call DOS as Subr
$CD4E STAT Term have Input?

$D400 FMS Initialization
$D403 FMS Close All Files
$D406 Execute FMS Funct'n

$D409-D40A FCB Base Pointer
$D40B-D40C Current FCB Address

$D435 FMS Verify Flag

------- Miscellaneous --------
$DFD0-DF Logic/Real Addr Table
$DFE0-E1 Control Port's Address
$DFE2 Terminal Echo Flag

- SBUG-E Compatible Addresses -

$F804 INCH Input Char fm Term
$F806 INCHE Input Char & Echo
$F808 INCHEK Ck Term: Inp Char?
$F80A OUTCH Send Char to Term
$F80C PDATA Print Char String
$F80E PCRLF Print CR & LF
$F810 PSTRNG PCRLF then PDATA
$F812 LRA Load Real Address

------ Interrupt Vectors -----

$FFF0	$DFCD	Reserved User Vec
$FFF2	$DFC2	SWI3 Service Routn
$FFF4	$DFC4	SWI2 Service Routn
$FFF6	$DFC6	FIRQ Service Routn
FFF8	$DFCB	IRQ Service Rout'n
$FFFA	$DFCA	SWI Service Rout'n
$FFFC	--	NMI (SBUG-E)
$FFFE	--	Reset (SBUG-E)

SIGNAL CONVENTIONS FOR RS232C INTERCONNECTIONS

Computer serial I/O controller interfaced as a Data Set (DCE)

RS232C Signal:	Prot Gnd	RxD	TxD	CTS	RTS	DTR	Sig Gnd	DCD	(4)	RxC	TxC	DSR
DB25P pin	1	2	3	4	5	6	7	8	11	15	17	20
Device: AAA Chicago "Elektra" DSP												
External	nc	in	out	nc	HI	HI	gnd	HI	--	--	--	--
Internal	nc	in	out	HI	nc	nc	gnd	HI	--	--	--	--
TERM con	16	15	14	13	12	11	10	9	--	--	--	--
Device: Data Systems '68' DSIO												
External	nc	in	out	in	out	--	--	in	--	--	--	--
Internal	nc	in	out	HI	nc	--	gnd	HI	--	--	--	--
PORT conn	--	3	2	5	4	--	1	6	--	--	--	--
Device: Southeastern Micro Systems DS-16												
External	gnd	+in	out	+in	out	nc	gnd	+in	nc	in	out	HI
Internal	nc	in	out	in	out	HI	gnd	in	nc	opt	TxC	nc
CEDB25 (2)	1	2	3	4	5	6	7	8	11	15	17	20
CEDA15 (3)	1	2	3	4	5	6	7	8	--	10	12	15

Computer serial I/O controller interfaced as a Data Terminal (DTE)

RS232C Signal:	Prot Gnd	TxD	RxD	RTS	CTS	DSR	Sig Gnd	DCD	(4)	TxC	RxC	DTR
DB25P pin	1	2	3	4	5	6	7	8	11	15	17	20
Device: AAA Chicago "Elektra" DSP												
External	nc	out	in	HI	nc	nc	gnd	in	--	--	--	out
Internal	nc	in	out	nc	in	nc	gnd	in	--	--	--	rts
MODEM con	16	15	14	13	12	11	10	9	--	--	--	7
Device: Data Systems '68' DSIO												
External	nc	out	in	out	in	--	gnd	in	--	--	--	--
Internal	nc	out	in	nc	HI	--	gnd	HI	--	--	--	--
PORT conn	--	2	3	4	5	--	1	6	--	--	--	--

Device: Southeastern Micro Systems DS-16

External / Internal	Gnd / nc	out / out	in / in	out / out	+in / in	HI / nc	gnd / gnd	+in / in	nc / nc	out / TxC	in / opt	HI / nc
CEDB25 (2)	1	3	2	5	4	20	7	8	11	17	15	6
CEDAI5 (3)	1	3	2	5	4	15	7	8	--	12	10	6

Notes:

1. Numbers shown in the above tables are pin numbers for the various connectors.
2. Card-edge contacts with DB25 connector installed.
3. Card-edge contacts with DA15 connector installed.
4. Undefined for standard RS232C service. Used for supervisory send data (SSD) by some devices.
5. I/O signals fitted with pullup resistors are denoted +in, +ou.
6. Active low signals are indicated by designation in lower case.
7. It is customary for a data terminal (DTE) to be fitted with a DB25P connector and for data communications equipment to be fitted with a DB25S connector. A standard RS232C cable therefore has a male (DB25P) connector at one end a female (DB25S) connector at the other.

Abbreviations and symbols used:

BUSY	Not shown above. RS232C pin 25. Some devices use this.
CTS	Clear to Send. Indicates receiver ready to accept data.
DB25P	D subminiature connector having 25 pins (male).
DB25S	D subminiature connector having 25 sockets (female).
DCD	Data Carrier Detected. Demodulator is receiving carrier.
DA15P	D subminiature connector having 15 pins (male).
DA15S	D subminiature connector having 15 sockets (female).
DSR	Data Set Ready. DCE (modem) ready for data.
DCE	Data Communications Equipment. Originally meant a modem, but with RS232C being adopted for use with computers, may indicate the computer itself when connected to DTE.
DTE	Data Terminal Equipment. Printer, CRT terminal. When two computers are connected, one must be designated as DTE, the other as DCE.
DTR	Data Terminal Ready. DTE ready for data.
External	Signal as presented to the off-board cable connector.
gnd	Ground.
HI	Signal pulled high through a pull up resistor.
in	Incoming signal, relative to the controller.
+in	Incoming signal pulled up through a resistor.
Internal	Signal as presented to the controller electronics.
nc	Not connected. Also shown as --.
out	Outgoing signal, relative to the controller.
+ou	Outgoing signal pulled up through a resistor.
Prot	Protective. Protective or frame ground.
RTS	Ready to Send. Indicates transmitter ready to send data.
RxC	Received Clock. Clock sent by the device sending RxD.
RxD	Received Data.
Sig	Signal. Used with gnd to designate signal ground.
SSD	Supervisory send data. Often has switchable sense (may be either active high or active low, selectable by hardware or software switch). Output by DTE, may be connected to CTS, DTR, DSR, DCD etc. as needed. Usually on pin 11.
TxC	Transmitted Clock. Clock sent by the device sending TxD.
TxD	Transmitted Data.

Connectors needed for some SS-30 serial boards:

SMS DS-16	Either DB25 or DA15 mounted on card edge. No holes. +12 Vdc is available for jumpering to the cable.
Elektra OPS	Sixteen pin DIP ribbon cable IDC, or wire direct.
DS"68" DSID	Ten pin Molex type KK, pins on 0.156 in. centers; or wire direct. Holes are available permitting installation of pullup resistors, if needed. +12 Vdc is supplied to pin 7 of the connector pad.

SIGNAL CONVENTIONS FOR RS232C INTERCONNECTIONS

Peripheral devices interfaced as a Data Terminal Equipment (DTE)

RS232C Signal:	Prot Gnd	TxD	RxD	RTS	CTS	DSR	Sig Gnd	DCD	(4)	TxC	RxC	DTR
DB25P pin	1	2	3	4	5	6	7	8	11	15	17	20

Device: Heath/Zenith H-19A, Z-19A video terminal 8250 ACE

External	Prot	out	in	out	in	in	Sig	in	---	---	---	out
Internal	Prot	out	in	out	in	in	Sig	rlsd	---	---	---	out

Device: Volker-Craig (Nabu) 4404 CHAT video terminal

External	Prot	out	in	out	in	nc	Sig	in	out	---	---	out
External	Prot	out	in	out	in	nc	Sig	in	out	---	---	out

Device: Southeastern Micro Systems ST-02 terminal board 6850 ACIA

External	Prot	out	in	out	+in	nc	Sig	nc	---	---	---	HI

Device: Okidata Microline 83A printer L S serial I/f

External	Prot	nc	in	HI	nc	in	Sig	---	out	---	---	out

Device: Concurrent Technology Hy-Type printer controller 6551 VIA

External	Prot	out	in	nc	---	---	Sig	---	---	---	---	out
Internal	??	out	in	nc	+in	in	??	in	---	---	---	out

Peripheral devices w/ Data Communications Equipment (DCE) interface

RS232C Signal:	Prot Gnd	RxD	TxD	CTS	RTS	DTR	Sig Gnd	DCD	(4)	RxC	TxC	DSR
DB25P pin	1	2	3	4	5	6	7	8	11	15	17	20

Device: Volker-Craig (Nabu) 4404 CHAT video terminal printer port

External	Prot	in	out	nc	HI	HI	Sig	HI	in	---	---	in

Notes:

1. Many data terminals (DTE) are fitted with a DB25P (male) connector. A standard RS232C terminal cable has a female (DB25S) connector at one end and a male (DB25P) at the other. Not all terminals follow this convention.
2. Active low signals are indicated by designation in lower case.
4. Undefined for standard RS232C service. Used for supervisory send data (SSD) by Okidata, SUPR by Nabu.
5. I/O signals fitted with pullup resistors are denoted +in, +ou.

Interface connectors used on video terminals

Heath/Zenith H/Z-19A video terminal
Terminal: DB-25P (DTE standard) 8250 ACE

Volker-Craig (Nabu) 4404 CHAT video terminal
Terminal: DB-25S (DCE standard with DTE signals)
Printer: DB-25S (DCE standard)

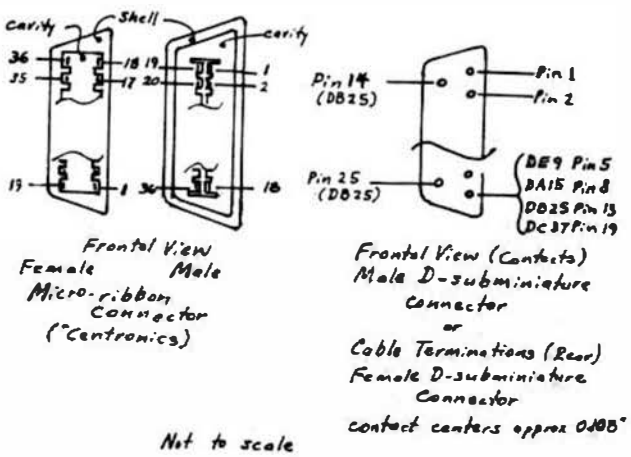
Southeastern Micro Systems ST-02 video terminal board
Terminal: DB25S specified, or wire direct to panel conn 6850 ACIA
Printer: DB25P specified, or wire direct to panel conn 6821 PIA

Okidata Microline 83A printer with L S serial I/f
Serial: DB-25P (DTE standard, with SSD added)
Parallel: 36-contact female micro-ribbon ("Centronics")

Concurrent Technology Hy-Type printer controller
Serial: DB-25P (DTE standard) 6551 VIA
Parallel: 36-contact female micro-ribbon ("Centronics") 6821 PIA

Connectors needed for parallel interface:

Manufacturer	Cable Connection	Shell	printer (supplied) female	cable printer end male
AMP Sp. In.				552274-1
Amphenol	Solder pot	metal	57-40360	57-30360
Amphenol	Ribbon cable IDC	metal	157-42360	157-32360
T&B Ansley	Ribbon cable IDC	plastic	609-36FA	609-36MA
TRW Cinch	Solder pot	metal	57-40360	57-30360
TRW Cinch	Ribbon cable IDC	metal	77-42360	77-32360



DEVELOPMENT TERMINAL PROGRAM

67 Fisher Road
Rochester, NY 14624
February 20, 1984

Mr. Don Williams
68 Micro Journal
PO Box 849
Hixson, TN 37343

Dear Don,

Although I've been a subscriber to "68 Micro Journal" since sometime in your first year of publication, this is the first time I've written you a letter. Despite my silence, I've enjoyed each and every issue and look forward to receiving many more. (I've also just bought Stylograph and am thus on a letter writing binge.) I also enjoyed the SS-50 section at the Philadelphia Personal Computer Show several years ago which I understand you arranged. Here's a belated thanks.

Down to business. On the enclosed disk along with this letter are three programs I've written which I've placed in the public domain. Although I've been playing in the 68XX world since early 1977, I didn't get disks and FLEX until about two years ago. These programs represent some of my efforts to learn to use FLEX and do something useful with it. The three programs are named DMP, BINCPY and DT.

DMP displays the contents of FLEX binary files, not like all of the other dump programs I've come across which dump disk blocks, but organized by FLEX binary records. This provides an easier way to analyze the structure of binary files.

BINCPY standing for Binary Copy will copy a FLEX binary file, squeezing out any holes left in the file by the FLEX APPEND command. This will occasionally find and remove enough empty space to cause a binary file to take up less disk blocks and can frequently be run to advantage on FLEX itself. It also provides the option of selectively copying or not copying individual binary records to the destination file. This feature has allowed me to extract portions of programs like the FLEX disk drivers, disassemble them, edit in changes like double sided operation and then merge the results back into the original program. The resulting program can then be squeezed down to size. This is much more efficient (and pleasing) than making modifications by appending patches.

The last and largest program, DT, standing for Development Terminal Program, is just that. It provides for terminal emulation on a FLEX system so the FLEX environment can be used to develop software for some other 68XX target machine which understands Motorola S1-S9 binary format. The DT program can then be used to download the binary program image to the target machine. It can also be used to upload memory images saving them as FLEX binary files and last but not least it can also save as a text file any "console dialogue" with the target machine.

All three programs are written in 6800/6809 compatible code so they should run on any normal FLEX machine with only the change of a few equates. The documentation within each source listing tells what changes must be made before the the programs are assembled. For the 6809, this assumes the use of the TSC assembler which understands 6800 opcodes. I consider myself fortunate to have come across the fine efforts of Leo Taylor and friends early on and I took his admonishment about 6800/6809 compatibility to heart. Please feel free to publish these listings, put them on your bulletin board or whatever you think appropriate. These programs have been available on the FLEXNET bulletin board for some time but I do not believe many of your readers are aware of it and long distance phone transfer can be expensive. One of FLEXNET's other remote users has asked me to get DT working on a Color Computer. Should this prove successful, I will send you a copy of the results. The problem is the Color Computer's bit banger serial interface (and I don't have a COCO). DT assumes 6850's.

Last but certainly not least, please keep up the good work. A lot of people out here are depending on you.

Peace,

J. Chris Hausler

Editor's Note: Thanks Chris for the nice and encouraging words. I miss the get-togethers like the Philly affair. However, the world kept spinning on and things changed. I don't believe that we will ever have things, as they used to be. Seems the older I get, the more I remember the 'good old days' and long to redo them again. But one thing is certain: because of loyal readers (who I feel towards as friends also) I still get enjoyment out of doing 68 Micro Journal.

Despite what some might believe, it is not the financial reward that keeps us going. Fact is, I and nearly everyone here could have done better elsewhere. I have turned down offers of employment (never really worked for anyone else, ever) with great sounding titles and pretty good money, mainly because I have honestly enjoyed my association with the magazine and all you loyal readers - FRIENDS! Because of my association with so many of you it enhanced my market potential, but I freely opted to continue running CPI (68 Micro Journal's parent corp) and have not regretted it one minute, despite my bank balance.

So I, like you, and probably thousands of other yearn for some more Philly type get-togethers. Maybe we can - what do most of the rest of you want?

DMW

- - -

```
*  DT - Development Terminal program.  The purpose
*  of this program is to provide a Flex based
*  development tool for single board or other small
*  68XX systems.  It supports those systems whose
*  primary mode of program loading/saving uses
*  Motorola S1 format ASCII hexadecimal coded
*  binary records.  All communications with the
*  system under development is through its console.
*  When this program is awakened, it shows a menu
*  of single letter commands and prompts for one of
*  them.  If any other character is entered, the
*  program responds by redisplaying the menu text.
*
*  The commands are:
*
*  T - Talk to the other machine.  This is the
*      programs dumb terminal mode of operation.
*      Once entered, the program stays in this mode
*      until the user enters a CNTL @ (null code)
*      at which point the program exits terminal
*      mode and returns to command mode.
*
*  U - Upload, this is the same as "T" with the
*      addition that all characters transmitted
*      from the machine under development are saved
*      in a memory buffer in the terminal machine.
*      In addition to escaping this mode when
*      a CNTL @ is entered, it will also exit to
*      command mode if the memory data buffer goes
*      full.  The size of this buffer is dependent
*      on how much memory is in your system.  It
*      ranges from low memory to Flex MEMEND.  If
*      your memory size is too small to hold what
*      you want to save in one piece, you can save
*      it in several pieces and then use APPEND and
*      my program BINCPY to combine and edit the
*      pieces.
```

```
*
*  E - Examine displays the contents of the memory
*      data buffer so you can be sure you have
*      captured what you want before you save it.
*      Honors CNTL S, CNTL Q, others cause exit.
*
*  B - Binary save converts any S1 format records
*      in the memory data buffer into Flex format
*      binary records and writes them to disk.
*      It prompts for the filename when you enter
*      the command.
*
*  S - Straight save just copies the memory data
*      buffer contents to a Flex text file.  It
*      prompts for the filename when you enter the
*      command.

      PAG
*  Z - Initializes the memory data buffer to empty.
*
*  D - Download prompts for the name of a Flex
*      format binary file you wish to download to
*      the machine under development.  It then
*      prompts for an offset to be added to the
*      addresses in the file if so desired.  If no
*      offset is desired, just respond with a <CR>.
*      The program will then read in the file,
*      convert it to S1 format records and send it
*      to the machine under development.  During
*      download, if the machine under development
*      echoes any data back to the terminal port,
*      this program will try to display it.
*      Characters may be occasionally missed or
*      garbaged however when the terminal program
*      is off reading from the disk.  The
*      download is however correct.  If you want to
*      terminate this function early, striking any
*      key on your terminal will abort the process
*      at the end of the next S1 record.  Note that
*      in this case, the terminal program does not
*      send an "S9" to the other machine, so you
*      will have to manually from U or T modes.
*
*  X - Exit just returns you to Flex.
*
*  Usage Examples:
*
*  To upload binary data, enter the U command,
*  then direct the machine under development to
*  "punch" using S1 format the range or ranges of
*  memory you wish to save.  (Note that it is not
*  necessary to only punch and save one range of
*  memory at a time, the B command can handle
*  multiple ranges.)  When this is complete,
*  type CNTL @ to exit U mode and type B to save
*  the data.  You will be prompted for a filename.
*  Enter the filename and the machine will then
*  save the data.
*
*  Downloading binary data is a single step
*  process.  Just follow the instructions for
*  the D command.  The offset option will allow
*  you to say assemble a program to burn into a
*  68701 MPU EPROM, offset download it to the
*  68701 where it will then transfer and burn it
*  itself.  (Using PROBUG or something similar.)
      PAG
*  Configuring this program.  In addition to
*  setting the FLEX equate dependent on whether
*  running on a 6800 or 6809, you must set the
*  equates TPORT and MPORT to match your hardware.
*  For this program to function correctly, both
*  ports must be 6850's and TPORT must be the same
*  port as the FLEX console port.  Have fun!
*
*  I place this program in the public
*  domain.  J. C. Hausler  27-FEB-83
*
*  DOS EQUATES - This routine is written
*  using only 6800 opcodes.  Changing
*  the equate below for FLEX from $C000
*  to $A000 should allow it to run on
*  a 6800 as well as a 6809.  At least
*  so they tell me.
*

C000  FLEX    EQU   $C000

CB40  FCB     EQU   FLEX+$B40
CC2B  MEMEND  EQU   FLEX+$C2B
CD03  WARMS   EQU   FLEX+$D03
CD15  GETCHR  EQU   FLEX+$D15
CD18  INBUFF  EQU   FLEX+$D18
CD1E  PSTRNG  EQU   FLEX+$D1E
CD24  PCRLF   EQU   FLEX+$D24
CD2D  GETFIL  EQU   FLEX+$D2D
CD33  SETEXT  EQU   FLEX+$D33
CD36  ADDBX   EQU   FLEX+$D36
CD42  GETHEX  EQU   FLEX+$D42
D403  FMSCLS  EQU   FLEX+$1403
D406  FMS     EQU   FLEX+$1406

      *  The following equate TPORT must point to
      *  the same device FLEX is using as a
      *  console device.  IMPORTANT NOTE:  THE
      *  BAUD RATE OF THE DEVELOPMENT MACHINE
      *  PORT MUST BE NO MORE THAN HALF THE BAUD
      *  RATE OF THE FLEX TERMINAL PORT FOR
      *  RELIABLE OPERATION.

E004  TPORT   EQU   $E004     FLEX TERMINAL PORT
E006  MPORT   EQU   $E006     DEVELOPMENT MACHINE PORT
            PAG

                  *  The storage areas start at low memory and
                  *  go up to FLEX MEMEND.

0000                    ORG   0

0000              CURADS  RMB   2       REAL BINARY FILE ADDRESS
0002              CURPNT  RMB   2       BINARY RECORD BUF POINTER
0004              STACK   RMB   2       STACK POINTER STORAGE
0006              BUFPNT  RMB   2       DATA BUFFER POINTER
0008              DATEND  RMB   2       DATA BUFFER INPUT POINTER
000A              TMPBUF  RMB   2       S1 REC DATA BUFF POINTER
000C              TMPIND  RMB   2       S1 REC DATA INPUT POINTER
000E              IHIGH   RMB   2       S1 RECORD ADDRESS
0010              TEMPI   RMB   2       TEMPORARY I
0012              CHKSUM  RMB   1       S1 RECORD CHECKSUM
0013              BYTECT  RMB   1       S1 RECORD BYTE COUNT
0014              OPNCOD  RMB   1       OPEN FILE CODE
0015              EXTCOD  RMB   1       FILE EXTENSION CODE
0016              SAVFLG  RMB   1       SAVING DATA TO BUFFER FLAG
```

```
                    * S1 record data buffer

0017                TMPBEG  RMB   128
            0097    TMPEND  EQU   *

                    * The following must be in order
                    * as it is the structure of a
                    * FLEX binary record.

0097                TYPE    RMB   1         RECORD TYPE
0098                SADS    RMB   2         START ADDRESS
009A                BCNT    RMB   1         BYTE COUNT
009B                DATBUF  RMB   256       DATA BUFFER

                    * Upload Data Buffer
                    * Ends at FLEX MEMEND

            0198    BUFBEG  EQU   *
                            PAG

                    * Main Program - User is vectored from here
                    * to the various processing routines as a
                    * response to his input.

C100                        ORG   FLEX+$100
C100 20  01         DT      BRA   START
C102 01             VN      FCB   1         VERSION NUMBER
C103 86  03         START   LDAA  #3        6850 MASTER RESET CODE
C105 B7  E006               STAA  MPORT     DO IT
C108 86  11                 LDAA  #$11      8 BITS 2 STOP BITS
C10A B7  E006               STAA  MPORT
C10D 8E  0198               LDX   #BUFBEG   START OF DATA BUFFER
C110 9F  08                 STX   DATEND    MARK BUFFER EMPTY

                    * Main Loop

C112 8E  0687       PROMEN  LDX   #MENUMS   DISPLAY COMMAND MENU
C115 BD  C81E               JSR   PSTRNG
C118 8E  C300       CMDMOD  LDX   #PROMPT   PROMPT STRING
C11B BD  C81E               JSR   PSTRNG
C11E BD  CD15               JSR   GETCHR    WELL?
C121 84  5F                 ANDA  #$5F      MASK FOR UC/LC
C123 81  55                 CMPA  #'U       UPLOAD DATA SAVE
C125 27  34                 BEQ   UPLOAD
C127 81  44                 CMPA  #'D       DOWNLOAD BINARY FILE
C129 27  28                 BEQ   DNLOAD
C12B 81  54                 CMPA  #'T       TALK TO MACHINE
C12D 27  2A                 BEQ   TRNVAL
C12F 81  42                 CMPA  #'B       S1 TO FLEX BINARY SAVE
C131 27  7E                 BEQ   BINSAV
C133 81  53                 CMPA  #'S       STRAIGHT BUFFER SAVE
C135 27  53                 BEQ   STSAVE
C137 81  45                 CMPA  #'E       SHOW BUFFER CONTENTS
C139 27  2A                 BEQ   SHOBUF
C13B 81  5A                 CMPA  #'Z       INITIALIZE BUFFER POINTER
C13D 27  10                 BEQ   ZEROBF
C13F 81  58                 CMPA  #'X       EXIT TO FLEX
C141 26  CF                 BNE   PRMENU    ? - PROMPT WITH MENU
C143 BD  0403               JSR   FMSCLS
C146 8E  C510               LDX   #BYEMSG   SAY GOODBYE
C149 BD  CD1E               JSR   PSTRNG
C14C 7E  C003               JMP   WARMS     AND LEAVE

                    * Mark data buffer empty

C14F 8E  0198       ZEROBF  LDX   #BUFBEG   START OF DATA BUFFER
C152 9F  08                 STX   DATEND    MARK BUFFER EMPTY
C154 20  C2                 BRA   CMDMOD

                    * Link to the D (Download) command

C156 7E  C338       DNLOAD  JMP   DOWNL9    LINK
                            PAG

                    * The U and T commands are the same except
                    * that the U command saves the entire
                    * "conversation" as received from the machine.

C159 86  00         TRNVAL  LDAA  #0        TURN OFF BUFFER SAVE
C15B 97  16         UPLOAD  STAA  SAVFLG    (ASCII U IS NON-ZERO)
C15D BD  ED24               JSR   PCRLF
```

```
C160 9E  08                 LDX   DATEND    POINTER TO DATA BUFFER
C162 7E  C410               JMP   LOOP

                    * The E command just displays the contents
                    * of the data buffer exactly as received
                    * from the machine under development.

C165 BD  CB24       SHOBUF  JSR   PCRLF
C168 8E  0198               LDX   #BUFBEG   START OF BUFFER
C16B 9C  08         SHLOOP  CPX   DATEND    AT END OF BUFFER
C16D 27  A9                 BEQ   CMDMOD    YES
C16F A6  84                 LDAA  0,X       GET CHARACTER
C171 BD  C457               JSR   TEROUT    DISPLAY IT
C174 30  01                 INX             POINT TO NEXT
C176 BD  C44B               JSR   TERMIN    CHECK FOR CHARACTER
C179 24  F0                 BCC   SHLOOP    NOPE
C17B 81  13                 CMPA  #$13      IS IT CNTL S
C17D 26  99                 BNE   CMDMOD    NO - EXIT
C17F BD  C44B       SHWAIT  JSR   TERMIN    YES - WAIT FOR CNTL Q
C182 24  FB                 BCC   SHWAIT
C184 81  11                 CMPA  #$11      IS IT CNTL Q
C186 26  90                 BNE   CMDMOD    NO - EXIT
C188 20  E1                 BRA   SHLOOP

                    * The S command writes the contents of the
                    * data buffer to a text disk file exactly
                    * as received from the machine under development.

C18A 86  02         STSAVE  LDAA  #2        OPEN FOR WRITE
C18C C6  01                 LDAB  #1        .TXT EXTENSION
C18E BD  C209               JSR   OPNFIL    OPEN FILE
C191 25  18                 BCS   ENDSAV    ERROR RETURN
C193 BE  0198               LDX   #BUFBEG   START OF DATA BUFFER
C196 9F  06                 STX   BUFPNT
C198 BD  C210       NXTCHR  JSR   GETDAT    GET ONE BYTE
C19B 27  0E                 BEQ   ENDSAV    END OF BUFFER REACHED
C19D 8E  C840               LDX   #FCB      FILE CONTROL BLOCK
C1A0 BD  D406               JSR   FMS       WRITE BYTE TO FILE
C1A3 27  F3                 BEQ   NXTCHR    GO GET NEXT
C1A5 8E  C528               LDX   #SEMRJT   ERROR ON WRITE
C1A8 BD  CD1E               JSR   PSTRNG    TELL HUMAN ABOUT IT
C1AB BD  D403       ENDSAV  JSR   FMSCLS    CLOSE FILE
C1AE 7E  C11B               JMP   CMDMOD    GO PROMPT FOR COMMAND
                            PAG

                    * The B command processes the data buffer as
                    * Motorola S1 records which would have been
                    * received in the data buffer as a response to a
                    * punch command to the machine under development.
                    * The resulting binary data is written to disk.

                    * Open the output file and initialize stuff

C1B1 86  02         BINSAV  LDAA  #2        OPEN FOR WRITE
C1B3 97  97                 STAA  TYPE      (ALSO SET RECORD TYPE)
C1B5 5F                     CLRB            .BIN EXTENSION
C1B6 BD  C209               JSR   OPNFIL    OPEN FILE
C1B9 25  5C                 BCS   ENDBIN    ERROR ON OPEN
C1BB 10DF 04                STS   STACK     SAVE STACK POINTER
C1BE 8E  0198               LDX   #BUFBEG   POINT TO DATA BUFFER
C1C1 9F  06                 STX   BUFPNT    SAVE POINTER
C1C3 8E  0000               LDX   #0        GET A ZERO FOR INITIALIZE
C1C6 BD  62                 BSR   INITBS    INITIALIZE SOME STUFF

                    * This first part of the B command processing
                    * retrieves the data from one S1 record and
                    * puts it in a buffer for processing by the
                    * second part of the B command (PUTBIN).

C1C8 8E  0017       NXTS1R  LDX   #TMPBEG   S1 DATA BUFFER START
C1CB 9F  0A                 STX   TMPBUF    SAVE POINTER
C1CD BD  4E         NXTBYT  BSR   GETDAT    GET ONE BYTE
C1CF 27  3C                 BEQ   EXITBS    END OF BUFFER REACHED
C1D1 81  53                 CMPA  #'S       IS IT AN "S"
C1D3 26  F8                 BNE   NXTBYT    NOPE - GET ANOTHER
C1D5 BD  46                 BSR   GETDAT    GET ANOTHER BYTE
C1D7 27  34                 BEQ   EXITBS    END OF BUFFER REACHED
C1D9 81  31                 CMPA  #'1       IS IT AN "S1"
C1DB 26  F0                 BNE   NXTBYT    NOPE - TRY AGAIN
C1DD 0F  12                 CLR   CHKSUM    YES! - GOT AN S1 RECORD
```

```
C1DF 8D   60          BSR   ADBYTE   GET BYTE COUNT HEX BYTE
C1E1 80   02          SUBA  #2       LESS ADDRESS BYTES
C1E3 97   13          STAA  BYTECT   SAVE BYTE COUNT
C1E5 8D   51          BSR   BLOADD   GET ADDRESS
C1E7 8D   58  DATAIN  BSR   ADBYTE   GET DATA BYTE
C1E9 0A   13          DEC   BYTECT   DECREMENT BYTE COUNT
C1EB 27   0F          BEQ   DONEIN   END OF S1 RECORD
C1ED 9E   0A          LDX   TMPBUF   SAVE DATA BYTE
C1EF 8C   0097        CPX   #TMPEND  BUFFER FULL?
C1F2 27   0E          BEQ   TOOBIG   YES
C1F4 A7   84          STAA  0,X
C1F6 30   01          INX
C1F8 9F   0A          STX   TMPBUF
C1FA 20   E8          BRA   DATAIN   GET NEXT BYTE
C1FC 0C   12  DONEIN  INC   CHKSUM   CHECK CHECKSUM
C1FE 26   07          BNE   ERRCHK   BAD CHECKSUM
C200 20   77          BRA   PUTBIN   GOOD CHECKSUM - SAVE DATA
                      PAGE

*  Error and completion processing for B command

C202 8E   C574  TOOBIG LDX   #TOOBIG  S1 TOO BIG MESSAGE
C205 20   03          BRA   ERRBSV   GO TELL HUMAN
C207 8E   C542  ERRCHK LDX   #CHKERR  BAD CHECKSUM MESSAGE
C20A BD   CD1E  ERRBSV JSR   PSTRNG   DISPLAY IT
C20D 0D   9A    EXITBS TST   BCNT     ANY DATA HANGING AROUND?
C20F 27   03          BEQ   FIXSTK   NO
C211 BD   C285        JSR   PUTREC   YES SAVE IT
C214 1DDE 04  FIXSTK  LDS   STACK    RECOVER STACK POINTER
C217 BD   0403  EXBDIN JSR   FRSCLS   CLOSE ANY OPEN FILES
C21A 7E   C118        JMP   CMDMOD   AND ASK FOR MORE

*  Get byte from data buffer

C21D 9E   06  GETDAT  LDX   BUFPNT   GET INPUT BUFFER POINTER
C21F 9C   08          CPX   DATEND   AT END OF BUFFER?
C221 27   04          BEQ   EXDDAT   YES - Z FLAG SET
C223 A6   84          LDAA  0,X      NO - GET CHARACTER
C225 30   01          INX            POINT TO NEXT
C227 9F   06          STX   BUFPNT   SAVE POINTER
C229 39               RTS            Z FLAG SHOULD BE 0

*  This subroutine initializes the control
*  variables for the FLEX format binary
*  record data buffer.

C22A 9F   98  INITBS  STX   SADS     SAVE BINARY START ADDRESS
C22C 9F   00          STX   CURADS   AND AS CURRENT ADDRESS
C22E BE   0098        LDX   #DATBUF  POINTER TO DATA BUFFER
C231 9F   02          STX   CURPNT   SAVE POINTER
C233 0F   9A          CLR   BCNT     CLEAR BYTE COUNTER
C235 39               RTS

C236 20   90  LNISIR  BRA   NITSIR   LINK
                      PAGE

*  The following subroutines for the B command
*  for processing S1 records were lifted from
*  Engineering Note 100.  (If you don't know
*  what Engineering Note 100 is you haven't
*  been around the 6811 world very long.)

*  Get address from S1 record

C238 8D   07  BLOADD  BSR   #BYTE
C23A 97   0E          STAA  IXHIGH   HIGH ORDER BYTE OF ADDRESS
C23C 8D   03          BSR   #BYTE
C23E 97   0F          STAA  IXHIGH+1 LOW ORDER BYTE OF ADDRESS
C240 39               RTS

*  Get two data byte from S1 record

C241 8D   15  ADBYTE  BSR   INHEX    GET ONE HEX CHAR.
C243 48               ASLA           SHIFT-
C244 48               ASLA           BYTE-
C245 48               ASLA           TO UPPER-
C246 48               ASLA           NIBBLE
C247 1F   8948        TAB            TO B REG
C24A 8D   0C          BSR   INHEX    GET ANOTHER HEX CHAR.
C24C 34   04 ABE0     ABA            OR THE TWO TOGETHER
C250 1F   8948        TAB            TO B REG
C253 DB   12          ADDB  CHKSUM   ADD TO CHECKSUM
C255 D7   12          STAB  CHKSUM   SAVE CHECKSUM
C257 39               RTS

*  Get hex character from S1 record

C258 8D   C3  INHEX   BSR   GETBAT   GET BYTE
C25A 27   18          BEQ   NODATA   END OF BUFFER REACHED
C25C 80   30          SUBA  #$30     ASCII NUMBER BASE
C25E 2B   0F          BMI   NOTHEX   TOO SMALL TO BE HEX
C260 81   09          CMPA  #$09     A NUMBER?
C262 2F   0A          BLE   GOTHEX   YES WE HAVE IT
C264 81   11          CMPA  #$11     NO - HEX LETTER?
C266 2B   07          BMI   NOTHEX   NO - TOO SMALL
C268 81   16          CMPA  #$16     TOO BIG?
C26A 2E   03          BGT   NOTHEX   YES
C26C 80   07          SUBA  #7       JUST RIGHT!
C26E 39               RTS
C26F 8E   C4CA  GOTHEX LDX   #BADHEX  BAD HEX CHARACTER FOUND
C272 20   96          BRA   ERRBSV
C274 8E   C4EC  NODATA LDX   #ENDMSG  END OF BUFFER ENCOUNTERED
C277 20   91          BRA   ERRBSV
                      PAGE

*  This second part of the B command processing
*  copies the data retrieved from the S1 records
*  into a FLEX format binary record and when it is
*  full or the address changes, writes it to disk.

C279 8E   0097  PUTBIN LDX   #TMPBEG  S1 DATA BUFFER
C27C 9F   0C          STX   TMPINX
C27E 9E   0E          LDX   IXHIGH   S1 START ADDRESS
C280 9C   00          CPX   CURADS   CURRENT BIN FILE ADDRESS
C282 26   22          BNE   NEWREC   NOT SAME - NEW RECORD
C284 9E   0C  NXTBAT  LDX   TMPINX   POINTER TO S1 DATA
C286 9C   0A          CPX   TMPBUF   END OF S1 DATA RECORD?
C288 27   AC          BEQ   LNISIR   YES - GO GET ANOTHER
C28A A6   84          LDAA  0,X      GET DATA BYTE
C28C 30   01          INX            POINT TO NEXT
C28E 9F   0C          STX   TMPINX   SAVE POINTER
C290 9E   02          LDX   CURPNT   CURRENT BIN DATA POINT
C292 A7   84          STAA  0,X      STORE DATA BYTE
C294 30   01          INX            INCREMENT POINTER
C296 9F   02          STX   CURPNT   SAVE IT
C298 9E   00          LDX   CURADS   GET CURRENT REAL ADDRESS
C29A 30   01          INX            INCREMENT IT
C29C 9F   00          STX   CURADS   SAVE IT
C29E 0C   9A          INC   BCNT     INCREMENT BYTE COUNTER
C2A0 96   9A          LDAA  BCNT     GET IT
C2A2 81   FD          CMPA  #$FD     AT MAXIMUM?
C2A4 26   DE          BNE   NXTBAT   NO - GET NEXT BYTE
C2A6 9F   00  NEWREC  STX   CURADS   SAVE NEW REAL ADDRESS
C2A8 0D   9A          TST   BCNT     ANY DATA IN BUFFER?
C2AA 27   04          BEQ   NOREC    NOPE
C2AC BD   07          BSR   PUTREC   WRITE RECORD TO DISK
C2AE 9E   00          LDX   CURADS   GET CURRENT REAL ADDRESS
C2B0 BD   C22A  NOREC JSR   INITBS   SET UP FOR NEXT
C2B3 20   CF          BRA   NXTBAT
                      PAGE

*  Put FLEX format binary record to disk.

C2B5 8E   0097  PUTREC LDX   #TYPE     POINTER TO RECORD START
C2B8 B6   9A          LDAA  BCNT     DATA BYTE COUNTER
C2BA CB   04          ADDB  #4       PLUS HEADER BYTES
C2BC A6   84  PTLOOP  LDAA  0,X      GET BYTE
C2BE 30   01          INX            POINT TO NEXT
C2C0 9F   10          STX   TEMPI    SAVE POINTER
C2C2 8E   C840        LDX   #FCB     FILE CONTROL BLOCK
C2C5 BD   9406        JSR   PUTS     WRITE BYTE
C2C8 26   04          BNE   EWRITE   ERROR RETURN
C2CA 9E   10          LDX   TEMPI    GET BACK POINTER
C2CC 5A               DECB           COUNT DOWN BYTES
C2CD 26   ED          BNE   PTLOOP   MORE TO DO?
C2CF 39               RTS
C2D0 8E   C520  EWRITE LDX   #WRERR   WRITE ERROR MESSAGE
C2D3 BD   CD1E        JSR   PSTRNG   DISPLAY IT
C2D6 7E   C214        JMP   FIXSTK   AND RETURN

*  This subroutine opens a FLEX file.  It is
*  entered with AC A containing either a 1 or
```

Update to ELEKTRA!

Package #1	Package #2	Package #3
2 MHz 6809 CPU Board Super Floppy Controller OS-9™ w/Edit. Asm. Debugger	2 MHz 6809 CPU Board Super Floppy Controller 4K Humbug™ and Star-Dos™	2 MHz 6809 CPU Board Super Floppy Controller (No software)
$695.00	**$675.00**	**$550.00**

ELEKTRA OS-9™ with Editor, Assembler, and Debugger **$250.00**
ELEKTRA STAR-DOS™ (Adaptation Guide: $50.00) **$75.00**
OS-9™ Super Modem Program by Epstein Associates **$100.00**

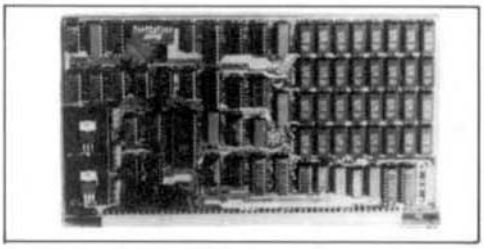
ELEKTRA Super Floppy Controller
(Supports SD, DD, SS, DS, 5", 8", 1MHz, 2MHz, 8 Drives
Emulates the DC-1, DC-2, DC-3, #28, #38, #48, #58 Controllers
Perfect upgrade for the DC-4 **$295.00**

Drivers for TSC's versions of **FLEX™** or **STAR-DOS™** (user installed) **$30.00**
OS-9™ Drivers (Reads and writes CoCo format too, user installed) **$50.00**

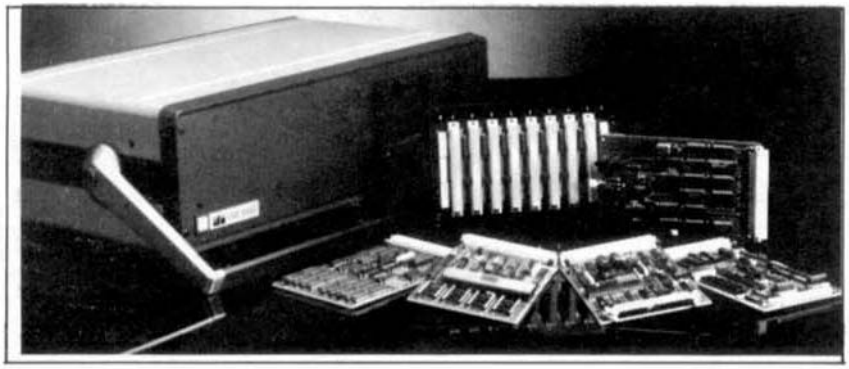
8" Floppy Drive Special w/manual, 90 day warranty
Siemens FDD 100-8 (SSDD) **$135.00**
Siemens FDD 200-8 (DSDD) **$185.00**

Removable Cartridge Winchester Drive (See next page for systems) **$1995.00**

2MHz Memory Boards with on board DAT
by Computer Excellence, Inc.
256K **$749.00** 512K **$1495.00**
1M **$2495.00**



Mizar 68000 — VME Development System with 256K RAM, 360K Floppy, 10 Mbyte Winchester, 4 Serial Ports (synchronous and/or asynchronous), OS-9, Screen Editor, Assembler, "C" compiler, and SASI interface **$6495.00**



Phone:
AAA Chicago Computer Center
Technical Consultation available most weekdays from 4 p.m. to 6 p.m. CST
(312) 459-0450 120 Chestnut Lane Wheeling, IL 60090
See our catalog and ordering information on the next page.

ELEKTRA COMPUTER SYSTEM Includes chassis, dual port serial interface with two cables, CPU 8/9, 4K Humbug, 56K static RAM, super floppy controller with inboard ribbon cable, Star-Dos, dual 80 track DSDD floppy drives (other combinations available; phone). OS-9 may be substituted for HUMBUG and STAR-DOS. $2795.00

ELEKTRA COMPUTER CABINET THE LARGEST SS-50 COMPUTER CABINET AVAILABLE! Made of heavyweight 0.090" thick aluminum. Interior is 18-1/2" wide by 21-7/8" deep by 6-3/4" high. Heavy duty A.C. line cord, A.C. fuse holder, EMI filter, Fan with filter. Back panel has 10 cutouts for 'D' type data connectors. Front panel has on/off power switch, 2 illuminated push button switches (Reset and NMI/Abort), and two cutouts for 2 full height or 4 half height 5-1/4" disk drives. $250.00

RACKMOUNT ELEKTRA COMPUTER CABINET 17"w x 21.5"d x 6.7"h. Holds one full or two half height 5-1/4" floppy drives. $250.00

Filler Plate for drive opening: $10.00 Fan Filter: $10.00

POWER SUPPLY Highest quality linear power supply CONSERVATIVELY rated at 15a @ 8v, 3a @ 16v, 3a @ -16v. Multi-tapped primary for fine tuning. $200.00

DISK REGULATOR BOARD WITH CABLES Standard version for 2 floppy drives $50.00
Heavy duty version for 1 Winchester drive and 1 floppy drive or 4 half heights $75.00

AUXILIARY POWER SUPPLY to power second Winchester drive. $125.00

ELEKTRA UNIVERSAL 59-50/59-50C MOTHERBOARD Heavyweight 0.125" thick, 18" long by 9" wide. 11 memory (50 pin) slots, 8 I/O (30 pin) slots. Complete address decoding and selection, as well as extended address capability, for I/O slots. Choice of 4, 8, or 16 addresses per I/O slot. 1" spacing between all memory and I/O slots. On board baud rate generator with low and high ranges providing jumper selectable rates of 75 through 9,600 for each of the five baud rate bits, as slow device circuitry permitting 1 Mhz 30 pin disk controllers to run with 2MHz 50 pin CPU boards.
Mounting hardware $5.00 Bareboard w/documentation: $80.00
Assembled w/tin connectors: $380.00 Assembled w/gold connectors: $460.00

ELEKTRA CHASSIS Includes cabinet, 110v power supply, power supply cables, standard disk regulator board with power cables, motherboard with gold square pin connectors, assembled and tested. (Add $25.00 for heavy duty regulator.) $950.00

ELEKTRA 2MHz CPU 8/9 Use either the 6802 or 6808 (to run 6800 software) or 6809. Has provision for up to 3 27/16 Eproms, 1K scratchpad, and MC6840 triple timer. Run OS-9™, FLEX™, STAR-DOS™ Bareboard: $50.00 Assembled: $275.00
Optional baud rate generator providing baud rates from 110 through 9,600 baud in two user selectable ranges. $25.00

ELEKTRA DPB DUAL PORT SERIAL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 or 16 addresses per port. RTS, CTS, DTR, DCD, IRQ, FIRQ/NMI, and baud rate can be appropriately implemented for each port.
Bareboard: $25.00 Assembled: $95.00
Cable with jack socket assemblies (two needed per board) Each: $25.00

ELEKTRA DPP DUAL PORT PARALLEL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 or 16 addresses per I/O slot. The direction of the TTL buffers can be controlled by either on board jumper connectors or by a signal from the peripherals. The interrupt request line for each port may be individually jumpered to either the IRQ or FIRQ/NMI bus line.
Bareboard: $25.00 Assembled: $80.00
Cable with jack socket assemblies (two needed per board) Each: $25.00

ELEKTRA 64K STATIC RAM/ROM MEMORY BOARDS with gold connectors (tin available) Assembled and tested. With 56K RAM $269.00 With 64K RAM $299.00

ELEKTRA UNIVERSAL SUPER FLOPPY CONTROLLER THE BEST 30 PIN FLOPPY DISK CONTROLLER THAT YOU CAN BUY! Controls up to four 5-1/4" drives and four 8" drives for a total of eight system drives. Single density or double density, 1MHz or 2MHz, 6800 or 6809. (Double density 8" requires 2MHz). Analog phase locked loop data separators with separate adjustments for 5" and 8" drives. Analog write precompensation circuit with separate adjustment for 5" and 8" drives. Designed to meet the data hold requirements of Western Digital floppy controller IC.
Bareboard (experts only): $100.00 Assembled and tested: $295.00
Disk with drivers, setup, and formatting utilities. Specify FLEX 2.0, 6800
Gen. FLEX, FLEX 9.0, FLEX 9.1, 6809 Gen FLEX or STAR-DOS; 5" or 8" $30.00
Disk with drivers for OS-9 (Specify 5" or 8") $50.00

ELEKTRA WINCHESTER SYSTEMS THE BEST WINCHESTER SYSTEMS THAT YOU CAN BUY! Has automatic error detection and CORRECTION of up to 11 bit burst errors. SS-50 bus, extended addressing capabilities, DMA, on board sector buffer, drivers included for 6809 FLEX, STAR-DOS, or OS-9. Specify whose version of FLEX that you are using. Drivers for 6800 FLEX2 are available for an additional $100.00. Price includes host interface, controller, drive(s), and cables.
7 Megabyte single drive sys. $1995.00 14 Megabyte dual drive sys. $2995.00
12 Megabyte single drive sys. $2295.00 24 Megabyte dual drive sys. $3595.00
19 Megabyte single drive sys. $2995.00 38 Megabyte dual drive sys. $4695.00
(19 Megabyte drives are the largest that can be supported by FLEX)
8 Megabyte removable cartridge single drive sys. $2995.00 Drive only $1995.00
Circuit boards, cables, software (No drive) 995.00
SS-50C DMA Bus interface board only 695.00

ELEKTRA HD-8 Cabinet for dual 8 1/4" floppy drives with power supply, linecord, fuse, power switch, and power cables to drives. 150.00

ELEKTRA HD-8W As above but with EMI filter, fan, and heavy duty power supply. Powers 1 floppy and 1 Winchester or 4 half height 5" floppies. 199.00

5" ribbon cable for dual outboard 5-1/4" disk drives 40.00
2" ribbon cable for dual inboard 5-1/4" disk drives 35.00
Custom cables available Phone

ELEKTRA HD-8 Dual 8" drive cabinet, EMI filter, fan with filter, power supply and power supply cables. 350.00
Filler plate 10.00
6' ribbon cable for dual 8" disk drives 45.00

ELEKTRA 30 PIN PROTOTYPING BOARD 20.00
ELEKTRA 50 PIN PROTOTYPING BOARD 40.00

GOLD 10 PIN CONNECTORS (Specify male with square pins or female) 1.50
TIN 10 PIN CONNECTORS (Specify male with square pins or female) .50

AAA CHICAGO COMPUTER CENTER (312) 459-0450
120 CHESTNUT LANE • WHEELING, IL 60090
Technical consultation available 4 PM to 6 PM most weekdays. Closed evenings and weekends.

TERMS Minimum order $20.00. Shipping and handling estimates within the Continental U.S., add 3% (MINIMUM $2.50). Illinois residents add 7% sales tax. We will refund your overestimated shipping and handling charges. Foreign shipping and handling, add 10% (MINIMUM $10.00). Foreign orders must be prepaid in U.S. dollars. Checks must be drawn on a U.S. bank. Heavy foreign items will be shipped air freight collect. Please phone between 4 PM and 6 PM weekdays if questions arise regarding shipping fees. Master Charge, Visa, and American Express honored.

Our apology. We are not staffed to answer technical inquiries through the mail. Please phone for technical help during the hours indicated above. The too frequent changing of our inventory and prices makes it uneconomical to publish a catalog. Our ads are intended to serve that purpose. Prices, specifications, and inventory are subject to change without advance notice.

SUPER MODEM PROGRAM Single character commands. No interrupts required. Transmit manually or transmit disk files (text) of any length to distant computer. Receive and save disk files (text) on local disk system. X-on/X-off supported. Tested for full duplex at speeds up to 9600 baud. Half duplex option. Echo option. Replaces CR with CR/LF (user option). Slow disk file transmit option.
Please specify 6800 or 6809, SSB, STAR-DOS™, or FLEX™, 5" or 8"
Instruction Manual and disk with both source and object code $75.00

OS-9 Super Modem Program by Epstein Associates with autodial, configuration file, etc. 100.00

ALL IN ONE
Editor — Text Processor — Mailing Labels — Mailing Lists — Multiple Form Letters
Use any CRT terminal and printer — Best Package For The Money Anywhere!
Specify 6800 or 6809, SSB, STAR-DOS™, or FLEX™, 5" or 8" 75.00
Add $35.00 for printed source listing, add $100 for source on disk.
All-In One, Write'n spell, and Spell'n Fix package 250.00

Software by Technical Systems Consultants, Inc.	FLEX™ Source (List)	FLEX™ Source (Disk)	FLEX™ Man. Only	FLEX™ Object w/Man.	UniFLEX™ Add. Man.	UniFLEX™ Man. Only	UniFLEX™ Object w/Man.
Gen. FLEX w/Edit & ASMB	—	—	—	250	—	—	—
FLEX 9.1 (DC-2) w/Edit & ASMB	—	—	25	150	40	100	550
Advanced Programmers Guide	—	—	25	—	—	—	—
Editor	100	250	25	50	—	—	—
Assembler	150	250	25	50	—	—	—
Debug	175	250	25	75	—	—	—
Extended Basic	—	—	25	100	20	50	200
Basic Precompiler	—	—	25	50	10	25	150
Sort/M	—	—	25	75	20	35	150
Utilities	—	Inc	25	75	10	25	150
Diagnostics	—	—	25	75	—	—	—
Text Processor	150	250	25	75	20	35	150
68000 X-ASMB on 6809	—	—	25	250	20	5	300
Pascal	—	—	50	00	25	50	300
Rel ASMB/Linking Loader	—	—	25	150	0	5	175
6800 X-ASMB on 6809	—	—	—	100	—	—	—
Cobol	—	—	—	—	30	75	750
Fortran 77	—	—	—	—	35	65	450

Software by Microware Systems Corp. (Suggested List Prices, varies/mfg)	Run-Time Package	Source	Manual Only	Object w/Man.
OS-9™ Level 1 w/Edit, Asm, Debug		400.00	40.00	250.00
OS-9™ Level 2 w/Edit, Asm, Debug		400.00	40.00	500.00
OS-9™ Edit, Asm, Debug Pkg			25.00	125.00
Device Driver for Disk Controller (Specify Model)		100.00		
Device Driver for ACIA and PIA		50.00		
Clock Driver for 6840 and 58167 clock chips		5.00		
Entertainment Pack I, or File Handler Toolbox, or NineCom, or Virtual Disk Driver (Level 2 only)			10.00	65.00
Print Spooler (Level 2 only)			15.00	95.00
RMA Relocatable Macro Assembler			20.00	125.00
RMA/68000 Cross Assembler				400.00
BASIC09™ W/Run-Time	50.00	N/A	25.00	200.00
BASIC09 Tour Guide Book			18.95	
"C" Compiler			25.00	250.00
C Programming Language (Kernighan & Ritchie)			19.95	
CIS Cobol compiler w/Forms 2 Prog. Gen.	50.00	N/A	40.00	400.00
Pascal Compiler	50.00	N/A	25.00	250.00
Sage Application Generator	300.00	N/A	25.00	995.00
Microware yearly support service (All products)				150.00
Edition Update w/ manuals 25.00		Version Update w/manuals		75.00

Special Software

STAR-DOS L1 (Specify ELEKT A or D -2, or DC-4)	$75.00	Adaptation guide		$50.00
2K MICROBUG 40.00	4K HUMBUG 75.00	Custom versions		$85.00
Spell'n Fix by Peter Stark 178.58	Write'n Spell by Peter Stark			75.11
All-In-One, Spell'n Fix, and Write'n Spell package				250.00
SUPER SLEUTH Disassembler System ($101.00 for OS-9 version)				99.00

SOLID DISK DRIVES	1 head Tandon	2 heads Tandon	2 heads CDC	1 head MPI	2 heads MPI
30 day guarantee					
5-1/4", 40 tracks	225.00	300.00	300.00	250.00	325.00
5-1/4", 80 tracks	300.00	375.00	375.00	325.00	400.00
MPI or CDC Service Manual (Specify 40 or 80 track)			25.00	Qume DT-8	550.00
Siemens 8" FDD 100-8 (SSDD) $1 5.00			FDD 200-8 (DSDD)		$185.00

SPECIAL BOARDS
Microtime II Tender and Clock Board (Assembled) 60.00
Data Mart 16K EPROM bareboard (2708 chips) 30.00

OUTBOARD EPROM PROGRAMMERS BY OPTIMAL TECHNOLOGY
Model EP-2A-79 (Personality modules extra) 169.00
Optimal Technology, Inc. 30 pin parallel I/O board for EP-2A-79 37.00
FLEX™ Software package for EP-2A-79 (Specify 6800 or 6809) 30.00
OS-9 Software package for EP-2A-79 10.00
Model EP-2B-87 (RS-232/20 MA, Motorola Imt, 8K buffer, 1200/9600 baud) 575.00
Model E -2B-88-4 (Copies 1 to 4 EPROMS) 550.00
Personality/Copy Modules for 2708, 2716, 27C16, 2732, 27C32, 2732A, 2758, MCM68764, MCM68766, 2764, 27C64, 2764A, 27128, 27128A, 27256, 27C256, 2508, 2516, 2532, 2564, 25128, 2816, 281 RB7C32, 8751, 38E70 $17 to $39

Smoke Signal Broadcasting
DCB-4A Double Density Controller Board for 5" and 8" with DOS 549.00
SCB-69 6809 CPU Board 399.00

OMIX CLEARANCE SALE	LIST	OUR PRICE		LIST	OUR PRICE
Cable (Per I/O)	24.95	20.00	6800 CPU board	224.03	100.00
80 X 24 Video Boards	398.76	250.00	Single prt ser, 1 cable	113.36	90.00
64 X 16 Video Boards	198.71	100.00	Dual prt par, 2 cables	138.32	110.00
16K Mem Bds, w/cntl reg		145.00	Systems (trade-ins)		Phone
93L422 DAT chip	17.50	15.00			

HELIX

64K 6809 Computer	$2398.00	Other computer systems available	
DMA 5" and 8" Floppy Controller	495.00	6809 CPU Board	495.00
68008 board for SS-50	595.00	CP/M-68K	350.00

COMPUTER EXCELLENCE
2MHz Memory board with on board DAT 256K $749.00 512K $1695.00 1M $2495.00

MIZAR
68000, VME development system with 256K RAM, 360K floppy, 10 Mbyte Winchester, 4 serial ports (synchronous and/or asynchronous), OS-9, screen editor, assembler, "C" compiler, and SASI interface $6495.00

SPECIALS

* SSB BFO Floppy Disk Controll (Version 3)		Run FLEX or SSB DOS	100.00
* SWTPC DC-4*	230.00	MP-MB (4K bareboard)	9.95
* SWTPC DMF-2*	595.00	S-32* AM not included	124.50
* SWTPC MP-09 2MHz CPU*	$295.00	*While supplies last	
* High speed tape reader	50.00	300 Baud acoustic modem	129.00
* TI 810 Printer w/lower case and full vertical forms control			1200.00

WARNING AAA Chicago Computer Center does not provide repair or diagnostic service for customer assembled boards. AAA Chicago Computer Center does warranty and maintain service for our assembled boards.

```
                    *  a 2 depending on whether the file is to be
                    *  opened for read or write respectively.
                    *  AC B should contain the desired default
                    *  file extension code.  The file is opened
                    *  in any case with space compression turned
                    *  off.

C2B9 97  14         OPNFIL  STAA    OPNCOD      1=READ 2=WRITE
C2BB D7  15                 STAB    EXTCOD      0=.BIN 1=.TXT
C2BD 8E  C47F               LDX     #FILMSG     PROMPT HUMAN FOR FILENAME
C2E0 BD  CB1E               JSR     PSTRNG
C2E3 BD  CD1B               JSR     INBUFF      GET FILENAME
C2E6 8E  CB40               LDX     #FCB        FILE CONTROL BLOCK
C2E9 BD  CD2B               JSR     SETFIL      PUT FILENAME INTO FCB
C2EC 25  15                 BCS     FILERR      ERROR RETURN
C2EE 96  15                 LDAA    EXTCOD      GET EXTENSION CODE
C2F0 BD  CD33               JSR     SETEXT      SET DEFAULT EXTENSION
C2F3 96  14                 LDAA    OPNCOD      GET FILE OPEN CODE
C2F5 A7  04                 STAA    0,X         INTO FCB
C2F7 BD  0406               JSR     FMS         OPEN FILE
C2FA 26  0C                 BNE     OPENER      ERROR RETURN
C2FC 86  FF                 LDAA    #$FF
C2FE A7  08 30              STAA    59,X        TURN OFF SPACE COMPRESSION
C301 4F                     CLRA                OK OPEN - CLEAR CARRY FLAG
C302 39                     RTS
C303 8E  C4A3       FILERR  LDX     #FORMER     FILENAME FORMAT ERROR
C306 20  03                 BRA     ERRMSG
C308 8E  C4B7       OPENER  LDX     #OPNERR     OPEN ERROR
C30B BD  CD1E       ERRMSG  JSR     PSTRNG      DISPLAY ERROR MESSAGE
C30E 43                     COMA                ERROR - SET CARRY FLAG
C30F 39                     RTS

                    *  Get byte routine

C310 9F  10         GETBYT  STX     TEMPX       HIDE X REG
C312 8E  CB40               LDX     #FCB
C315 BD  0406               JSR     FMS         GET ONE BYTE
C318 26  03                 BNE     ERREAD      ERROR RETURN
C31A 9E  10                 LDX     TEMPX       RESTORE X REG
C31C 39                     RTS
C31D 8E  C537       ERREAD  LDX     #SEREAD     READ ERROR
C320 BD  CD1E               JSR     PSTRNG
C323 10DE 04                LDS     STACK       RESTORE STACK POINTER
C326 20  0A                 BRA     ERRDWL

C328 86  53         ENDDWL  LDAA    #'S         PUT S9 TO MACHINE
C32A BD  C410               JSR     PUTBYT
C32D 86  39                 LDAA    #'9
C32F BD  C410               JSR     PUTBYT
C332 BD  0403       ERRDWL  JSR     FMSCLS      CLOSE FILE
C335 7E  C11B               JMP     CMDMOD      BACK FOR MORE

                    *  The D command reads a FLEX binary file,
                    *  converts it record by record into SX records
                    *  and sends them to the machine under development.
                    *  If the machine echos any of this data back to
                    *  the terminal machine (us), it is displayed on
                    *  the terminal.

                    *  Get filename and offset address from human

C338 86  01         DOWNLD  LDAA    #1          OPEN FOR READ
C33A 5F                     CLRB                .BIN EXTENSION
C33B BD  9C                 BSR     OPNFIL      OPEN FILE
C33D 25  F3                 BCS     ERRDWL      ERROR RETURN
C33F 10DF 04                STS     STACK       SAVE STACK POINTER
C342 8E  0000               LDX     #0          INITIALIZE OFFSET ADDRESS
C345 9F  0E                 STX     INIGH       TO ZERO
C347 8E  C491               LDX     #MESOFF     OFFSET PROMPT MESSAGE
C34A BD  CB1E               JSR     PSTRNG
C34D BD  CD1B               JSR     INBUFF      GET USER RESPONSE
C350 BD  CB42               JSR     GETHEX      CONVERT TO HEX
C353 25  02                 BCS     SILOOP      NO RESPONSE
C355 9F  0E                 STX     INIGH       SAVE OFFSET ADDRESS

                    *  Find next binary record

C357 8E  CB40       SILOOP  LDX     #FCB        FILE CONTROL BLOCK
C35A BD  0406               JSR     FMS         GET BYTE FROM FILE
C35D 26  C9                 BNE     ENDDWL      END OF FILE (PROBABLY)
C35F 81  02                 CMPA    #2          IS IT START OF RECORD
C361 27  0A                 BEQ     GOTREC      YES - GO PROCESS IT
C363 81  16                 CMPA    #$16        IS IT START ADDRESS
C365 26  F0                 BNE     SILOOP      NO - CONTINUE SEARCH
C367 BD  A7                 BSR     GETBYT      YES - READ START ADDRESS
C369 BD  A5                 BSR     GETBYT      - AND IGNORE IT
C36B 20  EA                 BRA     SILOOP      GO CONTINUE SEARCH

                    *  Read in binary record

C36D BD  A1         GOTREC  BSR     GETBYT      HIGH ORDER ADDRESS
C36F 97  98                 STAA    SADS
C371 BD  9D                 BSR     GETBYT      LOW ORDER ADDRESS
C373 97  99                 STAA    SADS+1
C375 BD  99                 BSR     GETBYT      BYTE COUNT
C377 97  9A                 STAA    BCNT
C379 1F  B94D               TAB                 BYTE COUNT TO B REG
C37C 8E  009B               LDX     #BATBUF     BINARY FILE DATA BUFF
C37F BD  8F         WRTGET  BSR     GETBYT      GET DATA BYTE
C381 A7  B4                 STAA    0,X         INTO BUFFER
C383 30  01                 INX                 POINT TO NEXT
C385 5A                     DECB                COUNT DOWN BYTES
C386 26  F7                 BNE     WRTGET      MORE TO GET
                            PAG

                    *  Add offset address to records start address

C388 96  99                 LDAA    SADS+1      START ADDRESS LOW BYTE
C38A 98  0F                 ADDA    INIGH+1     OFFSET ADDRESS LOW BYTE
C38C 97  99                 STAA    SADS+1      ACTUAL ADDRESS LOW BYTE
C38E 96  98                 LDAA    SADS        START ADDRESS HIGH BYTE
C390 99  0E                 ADCA    INIGH       OFFSET ADDRESS HIGH BYTE
C392 97  98                 STAA    SADS        ACTUAL ADDRESS HIGH BYTE

                    *  Calculate S1 record parameters

C394 8E  009B               LDX     #BATBUF     BINARY DATA AREA
C397 9F  02                 STX     CURPNT
C399 96  9A         NXTREC  LDAA    BCNT        #BYTES TO PROCESS
C39B 27  0A                 BEQ     SILOOP      NO MORE - GET NEXT RECORD
C39D C6  1B                 LDAB    #$1B        #DATA BYTES PER S1 RECORD
C39F 34  04 A0E0            SBA                 FROM BYTES REMAINING
C3A3 24  08                 BCC     NOTLST      MORE AFTER THIS
C3A5 34  04 ABE0            ABA                 SHORT RECORD, CALC LENGTH
C3A9 1F  B940               TAB                 TO B REG
C3AC 4F                     CLRA                ZERO BYTES REMAINING
C3AD 97  9A         NOTLST  STAA    BCNT        #BYTES REMAINING
C3AF D7  13                 STAB    BYTECT      #BYTES THIS S1 RECORD
C3B1 9E  98                 LDX     SADS        START ADDRESS THIS RECORD
C3B3 9F  00                 STX     CURADS      SAVE IT
C3B5 BD  CB36               JSR     ADDBX       CALC START FOR NEXT
C3B8 9F  98                 STX     SADS        SAVE IT
C3BA CB  03                 ADDB    #3          S1 RECORD BYTE COUNT
C3BC 34  04                 PSHB                HIDE IT

                    *  Put S1 record header to machine

C3BE 8E  C479               LDX     #HEADS1     (CR)(LF)(0)S1
C3C1 A6  84         NXTHDR  LDAA    0,X         GET MESSAGE CHAR.
C3C3 81  04                 CMPA    #4          IS IT EOT?
C3C5 27  06                 BEQ     PUTBCT      YES
C3C7 BD  47                 BSR     PUTBYT      NO - PUT TO MACHINE
C3C9 30  01                 INX                 POINT TO NEXT
C3CB 20  F4                 BRA     NXTHDR      AND DO IT
                            PAG

                    *  Put S1 record data to machine.

C3CD 0F  12         PUTBCT  CLR     CHKSUM      INITIALIZE CHECKSUM
C3CF 35  02                 PULA                GET S1 BYTE COUNT
C3D1 BD  22                 BSR     PUTHEX      PUT IT TO MACHINE
C3D3 96  00                 LDAA    CURADS      ADDRESS HIGH BYTE
C3D5 BD  1E                 BSR     PUTHEX
C3D7 96  01                 LDAA    CURADS+1    ADDRESS LOW BYTE
C3D9 BD  1A                 BSR     PUTHEX
C3DB 9E  02                 LDX     CURPNT      POINTER TO BINARY BUFFER
C3DD A6  84         NEXBAT  LDAA    0,X         GET BINARY DATA BYTE
C3DF BD  16                 BSR     PUTHEX
C3E1 30  01                 INX
C3E3 0A  13                 DEC     BYTECT      ANY MORE?
C3E5 26  F6                 BNE     NEXBAT      YES
C3E7 9F  02                 STX     CURPNT      NO SAVE BUFFER POINTER
```

```
C3E9 96 12          LDAA  CHKSUM    GET CHECKSUM
C3EB 43             COMA            1'S COMPLEMENT
C3EC BD 07          BSR   PUTHEX
C3EE BD 5B          BSR   TERRIN    DOES HUMAN WANT OUT
C3F0 24 A7          BCC   NITREC    NO
C3F2 7E C332        JMP   ERROML    YES

                    *  Put hex data to machine as two characters

C3F5 1F 894D  PUTHEX TAB             TO B FOR CHECKSUM
C3F8 DB 12          ADDB  CHKSUM    ADD TO CHECKSUM
C3FA D7 12          STAB  CHKSUM    SAVE UPDATED CHECKSUM
C3FC 34 02          PSHA            HIDE FOR 2ND BYTE
C3FE 44             LSRA            RIGHT-
C3FF 44             LSRA            JUSTIFY-
C400 44             LSRA            HIGH-
C401 44             LSRA            NIBBLE
C402 BD 04          BSR   OUTHIC    PUT OUT HIGH NIBBLE
C404 35 02          PULA            GET BACK BYTE
C406 84 0F          ANDA  #$F       ISOLATE LOW NIBBLE
C408 BB 30  OUTHIC  ADDA  #$30      ASCII NUMBER BASE
C40A 81 39          CMPA  #$39      IS IT A NUMBER
C40C 2F 02          BLE   PUTBYT    YES
C40E 8B 07          ADDA  #7        NO ITS A HEX LETTER

                    *  Put a byte to the machine

C410 9F 10  PUTBYT  STX   TEMPX     HIDE X REG
C412 BD 5A          BSR   MACOUT    SEND TO MACHINE
C414 BD 4C          BSR   MACHIN    ANYTHING FOR US
C416 24 02          BCC   EXITBT    NO
C418 BD 3B          BSR   TEROUT    YES DISPLAY IT
C41A 9E 10  EXITBT  LDX   TEMPX     RESTORE X REG
C41C 39             RTS
                    PAG

                    *  Main processing for U and T commands.
                    *  The program will stay in this loop until exited
                    *  by a CNTL @ or when in U the buffer goes full.

C41D BD 2C 1  IOLOOP BSR  TERRIN    CHARACTER FROM TERMINAL?
C41F 24 06          BCC   MCHINE    NO
C421 81 00          CMPA  #0        YES - IF CNTRL @ EXIT
C423 27 21          BEQ   EXITIO
C425 BD 47          BSR   MACOUT    PUT CHAR TO MACHINE
C427 BD 39  MCHINE  BSR   MACHIN    CHARACTER FROM MACHINE?
C429 24 F2          BCC   IOLOOP    NO
C42B BD 3A          BSR   TEROUT    YES PUT IT TO TERMINAL
C42D 0D 16          TST   SAVFLG    SAVING DATA TO BUFFER?
C42F 27 EC          BEQ   IOLOOP    NO
C431 A7 84          STAA  0,X       YES
C433 30 01          INX             POINT TO NEXT
C435 9F 08          STX   DATEND    SAVE CURRENT END OF BUFFER
C437 BC CC2B        CPX   #MEMEND   AT END OF BUFFER AREA?
C43A 26 E1          BNE   IOLOOP    NO
C43C 30 1F          DEX             BACK OFF ONE
C43E 9F 08          STX   DATEND
C440 8E C35B        LDX   #OOMESS   OUT OF MEMORY MESSAGE
C443 BD CB1E        JSR   PSTRNG
C446 0F 16  EXITIO  CLR   SAVFLG    TURN OFF SAVE FLAG
C448 7E C11B        JMP   CNDMOD    AND RETURN
                    PAG

                    *  These routines directly address the actual
                    *  I/O ports without going through FLEX.  They
                    *  assume that the ports are 6850's with the
                    *  control and status register the lower and
                    *  the data register the higher of the two
                    *  addresses.  If your machine is different,
                    *  you'll have to modify this routine.
                    *  If you do so remember that your TERMIN and
                    *  MACHIN must affect the C flag the same way.
                    *  The two port addresses MPORT and TPORT must
                    *  be set up to match your hardware and
                    *  software.  Also remember that TPORT must
                    *  be the same port as the FLEX console.

                    *  Get character from terminal

C44B F6 E004  TERRIN LDAB  TPORT    GET TERMINAL CSR
C44E 56              RORB           RORS TO C
C44F 24 05           BCC   TERMEX   RORF NOT SET
C451 B6 E005         LDAA  TPORT+1  RORS SET - GET CHARACTER
C454 84 7F           ANDA  #$7F     FORCE ASCII
C456 39       TERMEX RTS            AND LEAVE

                    *  Put character to terminal

C457 F6 E004  TEROUT LDAB  TPORT    GET TERMINAL CSR
C45A 56              RORB           TDRE-
C45B 56              RORB           TO C
C45C 24 F9           BCC   TEROUT   NOT SET - WAIT FOR IT
C45E B7 E005         STAA  TPORT+1  PUT CHARACTER TO TERMINAL
C461 39              RTS            AND EXIT

                    *  Get character from machine

C462 F6 E006  MACHIN LDAB  MPORT    GET MACHINE CSR
C465 56              RORB           RORS TO C
C466 24 09           BCC   MACHEX   NOT SET - EXIT
C468 B6 E007         LDAA  MPORT+1  GET CHARACTER
C46B 84 7F           ANDA  #$7F     FORCE ASCII
C46D 39       MACHEX RTS

                    *  Put character to machine

C46E F6 E008  MACOUT LDAB  MPORT    GET MACHINE CSR
C471 56              RORB           TDRE-
C472 56              RORB           TO C
C473 24 F9           BCC   MACOUT   NOT SET - WAIT FOR IT
C475 B7 E007         STAA  MPORT+1  PUT CHARACTER
C478 39              RTS
                     PAG

                    *  STRINGS AND THINGS

C479 0D 0A 00 53  HEADS1 FCB   $D,$A,0,'S,'1,4
C47D 31 04
C47F 45 4E 54 45  FILMSG FCC   'ENTER FILENAME: '
C483 52 20 46 49
C487 4C 45 4E 41
C48B 4D 45 3A 20
C48F 20
C490 04                  FCB   4
C491 41 44 44 52  MESOFF FCC   'ADDRESS OFFSET: '
C495 45 53 53 20
C499 4F 46 46 53
C49D 45 54 3A 20
C4A1 20
C4A2 04                  FCB   4
C4A3 42 41 44 20  FORMER FCC   'BAD FILENAME FORMAT'
C4A7 46 49 4C 45
C4AB 4E 41 4D 45
C4AF 20 46 4F 52
C4B3 4D 41 54
C4B6 04                  FCB   4
C4B7 4F 50 45 4E  OPNERR FCC   'OPEN ERROR ON FILE'
C4BB 20 45 52 52
C4BF 4F 52 20 4F
C4C3 4E 20 46 49
C4C7 4C 45
C4C9 04                  FCB   4
C4CA 49 4C 4C 45  BADHEX FCC   'ILLEGAL HEX CHARACTER ENCOUNTERED'
C4CE 47 41 4C 20
C4D2 48 45 58 20
C4D6 43 48 41 52
C4DA 41 43 54 45
C4DE 52 20 45 4E
C4E2 43 4F 55 4E
C4E6 54 45 52 45
C4EA 44
C4EB 04                  FCB   4
C4EC 45 4E 44 20  ENDMSG FCC   'END OF BUFFER FOUND IN SI RECORD'
C4F0 4F 46 20 42
C4F4 55 46 46 45
C4F8 52 20 46 4F
C4FC 55 4E 44 20
C500 49 4E 20 53
C504 31 20 52 45
C508 43 4F 52 44
```

```
C50C 04                 FCB   4
C500 3D 3E        PROMPT FCC   '=>'
C50F 04                 FCB   4
C510 54 48 20 54  BYEMSG FCC   /TH TH TH THAT'S ALL FOLKS'/
C514 48 20 54 48
C518 20 54 48 41
C51C 54 27 53 20

C520 41 4C 4C 20
C524 46 4F 4C 4B
C528 53 21
C52A 04                 FCB   4
C52B 57 52 49 54  SERRIT FCC   'WRITE ERROR'
C52F 45 20 45 52
C533 52 4F 52
C536 04                 FCB   4
C537 52 45 41 44  SEREAD FCC   'READ ERROR'
C53B 20 45 52 52
C53F 4F 52
C541 04                 FCB   4
C542 53 31 20 52  CHKERR FCC   'S1 RECORD CHECKSUM ERROR'
C546 45 43 4F 52
C54A 44 20 43 48
C54E 45 43 4B 53
C552 55 4D 20 45
C556 52 52 4F 52
C55A 04                 FCB   4
C55B 44 41 54 41  OOMESS FCC   'DATA STORAGE BUFFER FULL'
C55F 20 53 54 4F
C563 52 41 47 45
C567 20 42 55 46
C56B 46 45 52 20
C56F 46 55 4C 4C
C573 04                 FCB   4
C574 53 31 20 52  TOOLGM FCC   'S1 RECORD TOO LONG'
C578 45 43 4F 52
C57C 44 20 54 4F
C580 4F 20 4C 4F
C584 4E 47
C586 04                 FCB   4
C587 44 45 56 45  MENUMS FCC   'DEVELOPMENT TERMINAL COMMANDS ARE:'
C58B 4C 4F 50 4D
C58F 45 4E 54 20
C593 54 45 52 4D
C597 49 4E 41 4C
C59B 20 43 4F 4D
C59F 4D 41 4E 44
C5A3 53 20 41 52
C5A7 45 3A
C5A9 00 0A 00 0D         FCB   0D,$A,0,$D,$A,0
C5AD 0A 00
C5AF 20 20 54 20         FCC   '  T - TALK TO OTHER MACHINE'
C5B3 2D 20 54 41
C5B7 4C 4B 20 54
C5BB 4F 20 4F 54
C5BF 48 45 52 20
C5C3 4D 41 43 48
C5C7 49 4E 45
C5CA 0D 0A 00            FCB   $D,$A,0
C5CD 20 20 55 20         FCC   '  U - TALK TO MACHINE AND SAVE DATA'
C5D1 2D 20 54 41
C5D5 4C 4B 20 54

C5D9 4F 20 4D 41
C5DD 43 48 49 4E
C5E1 45 20 41 4E
C5E5 44 20 53 41
C5E9 56 45 20 44
C5ED 41 54 41
C5F0 0D 0A 00            FCB   $D,$A,0
C5F3 20 20 45 20         FCC   '  E - EXAMINE BUFFER CONTENTS'
C5F7 2D 20 45 58
C5FB 41 4D 49 4E
C5FF 45 20 42 55
C603 46 46 45 52
C607 20 43 4F 4E
C60B 54 45 4E 54
C60F 53
C610 0D 0A 00            FCB   $D,$A,0
C613 20 20 42 20         FCC   '  B - SAVE S1 DATA AS BINARY FILE'
C617 2D 20 53 41
C61B 56 45 20 53
C61F 31 20 44 41
C623 54 41 20 41
C627 53 20 42 49
C62B 4E 41 52 59
C62F 20 46 49 4C
C633 45
C634 0D 0A 00            FCB   $D,$A,0
C637 20 20 53 20         FCC   '  S - SAVE DATA AS TEXT FILE'
C63B 2D 20 53 41
C63F 56 45 20 44
C643 41 54 41 20
C647 41 53 20 54
C64B 45 58 54 20
C64F 46 49 4C 45
C653 0D 0A 00            FCB   $D,$A,0
C656 20 20 5A 20         FCC   '  Z - INITIALIZE BUFFER TO EMPTY'
C65A 2D 20 49 4E
C65E 49 54 49 41
C662 4C 49 5A 45
C666 20 42 55 46
C66A 46 45 52 20

C66E 54 4F 20 45
C672 4D 50 54 59
C676 0D 0A 00            FCB   $D,$A,0
C679 20 20 44 20         FCC   '  D - DOWNLOAD BINARY FILE'
C67D 2D 20 44 4F
C681 57 4E 4C 4F
C685 41 44 20 42
C689 49 4E 41 52
C68D 59 20 46 49
C691 4C 45
C693 00 0A 00            FCB   $D,$A,0
C696 20 20 58 20         FCC   '  X - EXIT TO FLEX'
C69A 2D 20 45 58
C69E 49 54 20 54

C6A2 4F 20 46 4C
C6A6 45 58
C6A8 0D 0A 00            FCB   $D,$A,0
C6AB 20 4F 54 48         FCC   '  OTHER - DISPLAY THIS MENU'
C6AF 45 52 20 2D
C6B3 20 44 49 53
C6B7 50 4C 41 59
C6BB 20 54 48 49
C6BF 53 20 4D 45
C6C3 4E 55
C6C5 0D 0A 00            FCB   $D,$A,0
C4CB 43 4E 54 4C         FCC   'CNTL B - EXIT U AND T MODES'
C6CF 20 42 20 2D
C6D3 20 45 58 49
C6D7 54 20 55 20
C6DB 41 4E 44 20
C6DF 54 20 4D 4F
C6E3 44 45 53
C6E3 0D 0A 04            FCB   $D,$A,4
                         END   DT
```

0 ERROR(S) DETECTED

SYMBOL TABLE:

```
A0B3F C036  BADMEI C4CA  BCUT   009A  BINSAV C1B1  BLDABO C23B
BUFBE6 019B BUFPNT 0006  BYEMSG C510  BYTECT 0013  CHKERR C542
CHKSUM 0012 CMDMOD C11B  CMDADS 0000  CURPNT 0002  DATAIN C1E7
DATBUF 0098 DATEND 0008  DBLOAD C156  DONEIN C1FC  DOUBLB C33B
DT     C100 ENDBIN C217  ENDDAT C229  ENDDWL C328  ENDMSG C4EC
ERDSAV C1A8 ERRDSV C20A  ENRCMD C207  ERRDWL C332  ERREAD C31D
ERRMSG C50B EWRITE C2D0  EXITDS C20E  EXITDT C41A  EXITIO C446
EXTCOD 0015 FCB    CB40  FILERR C303  FILMSG C47F  FIRSTA C214
FLEX   C000 FMS    D406  FMSCLS D402  FORMER C4A2  GETBYT C310
GETCHR CB15 GETDAT C21D  GETFIL C02D  GETMEI C042  GOTMEI C26E
HOTREC C36D HEADSI C479  HEIOAT C30D  INBUFF CB1E  INMEI  C25B
INITBS C22A IOLOOP C41D  LWISIR C236  MACMEI C46B  MACMIN C462
MACOUT C46E MCMINE C427  MENEND CC2B  MENUMS C587  MESOFF C491
MPDR?  ED06 MEMREC C2A6  MODATA C274  MOREC  C2BD  MOTMEI C2AF
MOTLST C3AD MITBYT C1CD  MITCHR C198  MITDAT C2B4  MI76ET C37F
MITNBR C3C1 MITREC C399  MITSIR C1C8  OOMESS C55B  OPENER C30B
OPMCOD 0014 OPMERR C4B7  OPNFIL C2B9  OUTHIC C40B  PCRLF  CD2A
PRMEMU C112 PROMPT C500  PSTRNG CD1E  PTLOOP C2BC  PUTDCT C3CD
PUTBIN C279 PUTBYT C410  PUTMEI C3F5  PUTREC C2B5  RDBYTE C241
```

```
SILOOP C357   SADS   0098   SAVFLG 00:e   SEREAD C537   SETEIT CD33
SEWAIT C52B   SWLOOP C1AB   SWORUF C165   SWWAIT C17F   STACK  0004
START  C103   STSAVE C18A   TEMPX  0010   TERMEX C456   TERMIN C44B
TERCUT C457   TMPBEB 0017   TMPBUF 000A   TMPEND 0097   TMPINB 000C
TOBIGH C574   TOOBIG C202   TPORT  E004   TRMBAL C159   TYPE   0097
UPLOAD C158   VN     C102   WARMS  C903   XHIGH  000E   ZEROBF C14F
```

```
*
*   BINCPY - Binary file copy utility.
*   This program has two modes of
*   operation.  This first is a straight
*   logical copy of the file and the
*   second is a prompted copy.  Both modes
*   pack the binary records, recovering
*   disk space lost due to the FLEX
*   APPEND utility.  This can frequently
*   result in less sectors required to
*   store a binary file.  It can be run to
*   frequent advantage on FLEX itself.
*   In prompted copy, the user is prompted
*   for each binary record.  At the prompt
*   he has the following options:
*
*      C - Copy the record shown and go on
*          to the next.
*      N - Don't copy the record shown and
*          go on to the next.
*      D - Display the contents of the
*          record shown and reprompt for
*          the same record.
*      E - Exit the program.  The output
*          file will contain only those
*          records already copied.
*      H - Help command.  Lists these
*          commands.
*
*   The user activates the desired option
*   by just striking the appropriate key.
*   A <CR> is not necessary.  Striking an
*   incorrect key will just reprompt for
*   the same record.  Both kinds of
*   records, data records and start
*   address records are shown.  Note that
*   you cannot Display a start record as
*   there is no data in a start record
*   other than the address itself  The
*   address is shown in the prompt.
*
*   The command to run the utility is:
*
*   BINCPY <input filename>,<output
*    filename>,P
*
*   The extension defaults to ".BIN".
*   The third argument, the "P", specifies
*   if present the prompted copy.  If not
*   present, the utility copies (and packs
*   if possible) the entire binary file.
*   If the "P" is present in the command
*   line as the third argument, the
*   prompting takes place.  Anything else
*   in the third argument field will cause
*   and error to be displayed and the
*   utility to exit.
*
*   I place this program in the public
*   domain.  J. C. Hausler  02-JAN-83
*
*   DOS EQUATES - This routine is written
*   using only 6800 opcodes.  Changing
*   the equate below for FLEX from $C000
*   to $A000 should allow it to run on
*   a 6800 as well as a 6809.  At least
*   so they tell me.
*
```

```
C000  FLEX   EQU   $C000
CD03  WARMS  EQU   FLEX+$C303
CD15  GETCHR EQU   FLEX+$CD15
CD18  PUTCHR EQU   FLEX+$CD18
CD1E  PSTRNG EQU   FLEX+$CD1E
CD24  PCRLF  EQU   FLEX+$CD24
CD27  MATCH  EQU   FLEX+$CD27
CD2D  GETFIL EQU   FLEX+$CD2D
CD33  SETEXT EQU   FLEX+$CD33
CD36  ADDBX  EQU   FLEX+$CD36
CD3C  OUTHEX EQU   FLEX+$CD3C
CD3F  RPTERR EQU   FLEX+$CD3F
CD45  OUTADR EQU   FLEX+$CD45
B403  FRSCLS EQU   FLEX+$1403
B406  FRS    EQU   FLEX+$1406
CB40  FCBIN  EQU   FLEX+$0B40
                   *
C100               ORG   FLEX+$0100
                   *
                   ****PROGRAM START
                   *
C100               BINCPY
C100 7E  C355      JMP   START
C103 01            VN    FCB   1
                   *
                   ****TEMPORARY STORAGE
                   *
C104               EADS  RMB   2       END ADDRESS
C106               NOPMPT RMB  1       PROMPT FLAG
C107               TEMPX RMB   2       X-REG TEMPORARY
C109               TSADS RMB   2       TEMP START ADDR
C10B               BUFPNT RMB  2       BUFFER POINTER
C10D               BPOINT RMB  2       BUFFER POINTER
C10F               TBCNT RMB   1       TEMP BYTE COUNT
C110               LINC  RMB   1       LOCATION COUNTER
C111               FCBOUT RMB  320     OUTPUT FCB
                   *
                   *  The following must be in order
                   *  as it is the structure of a
                   *  FLEX binary record.
                   *
C251               TYPE  RMB   1       RECORD TYPE
C252               SADS  RMB   2       START ADDRESS
C254               BCNT  RMB   1       BYTE COUNT
C255               DATBUF RMB  256     DATA BUFFER
                   *
                   ****PROCESS COMMAND LINE
                   *
                   *  OPEN INPUT FILE
                   *
C355               START
C355 8E  CB40      LDX   #FCBIN
C358 BD  CD2D      JSR   GETFIL
C35B 25  3F        BCS   CMDERR
C35D 4F            CLRA              DEFAULT .BIN EXT
C35E BD  CD33      JSR   SETEXT
C361 86  01        LDAA  #1          OPEN FOR READ
C363 A7  B4        STAA  0,X
```

```
C365 BD  0406          JSR   FMS
C368 26  37            BNE   OPRDER
C36A 86  FF            LDAA  #$FF
C36C A7  88 3B         STAA  59,X
                *
                *  OPEN OUTPUT FILE
                *
C36F CE  C111          LDX   #FCBOUT
C372 BD  CD2D          JSR   GETFIL
C375 25  25            BCS   CMDERR
C377 4F                CLRA                  DEFAULT .BIN EXT
C378 BD  CD33          JSR   SETEXT
C37B 86  02            LDAA  #2              OPEN FOR WRITE
C37D A7  84            STAA  0,X
C37F BD  0406          JSR   FMS
C382 26  22            BNE   OPWRER
C384 86  FF            LDAA  #$FF
C386 A7  88 3B         STAA  59,X
                *
                *  DETERMINE WHETHER TO PROMPT COPY
                *
C389 BD  CD27          JSR   MATCH
C38C 25  09            BCS   TERM
C38E 81  50            CMPA  #'P
C390 26  0A            BNE   CMDERR
C392 B7  C106          STAA  NOPMPT
C395 20  14            BRA   MAINLP
C397                TERM
C397 7F  C106          CLR   NOPMPT
C39A 20  0F            BRA   MAINLP
                *
                *  INPUT ERROR PROCESSING
                *
C39C                CMDERR
C39C 8E  C5C9          LDX   #EINPUT
C39F 20  60            BRA   ERRMSG
C3A1                OPRDER
C3A1 8E  C5DE          LDX   #EOPWRD
C3A4 20  5B            BRA   ERRMSG
C3A6                OPWRER
C3A6 8E  C5F4          LDX   #EOPWR
C3A9 20  56            BRA   ERRMSG
                *
                ****MAIN PROGRAM LOOP
                *
                *  LOOK FOR START OF RECORD
                *
C3AB                MAINLP
C3AB 8E  C840          LDX   #FCBIN
C3AE BD  0406          JSR   FMS
C3B1 26  4B            BNE   ENDFIL
C3B3 81  02            CMPA  #$02
C3B5 27  04            BEQ   STTYPE
C3B7 81  16            CMPA  #$16
C3B9 26  F0            BNE   MAINLP
C3BB                STTYPE
C3BB B7  C251          STAA  TYPE
                *
                *  READ IN RECORD AND CHECK FOR
                *  PROMPTED COPY
                *
C3BE BD  C4D0          JSR   GETREC
C3C1 7D  C106          TST   NOPMPT
C3C4 27  1D            BEQ   OUTDAT

                *
                *  PROMPT HUMAN FOR ACTION AND
                *  SHOW RECORD ADDRESS AND SIZE
                *
C3C6                DISPLP
C3C6 BD  42            BSR   DISPLY
C3C8 BD  CD15          JSR   GETCHR
C3CB 84  5F            ANDA  #$5F            MASK FOR UC/LC
C3CD 81  43            CMPA  #'C             COPY RECORD
C3CF 27  12            BEQ   OUTDAT
C3D1 81  4E            CMPA  #'N             DON'T COPY
C3D3 27  06            BEQ   MAINLP
C3D5 81  45            CMPA  #'E             EXIT PROGRAM NOW
C3D7 27  2B            BEQ   CLOSE
C3D9 81  44            CMPA  #'D             SHOW DATA RECORD
C3DB 27  0B            BEQ   DISDATA
C3DD 81  48            CMPA  #'H             HELP COMMAND
C3DF 27  12            BEQ   HELPME
C3E1 20  E3            BRA   DISPLP
                *
                *  WRITE RECORD
                *
C3E3                OUTDAT
C3E3 BD  C518          JSR   PUTREC
C3E6 20  C3            BRA   MAINLP
                *
                *  DISPLAY DATA IN RECORD
                *
C3EB                DISDATA
C3EB B6  C251          LDAA  TYPE
C3EB 81  02            CMPA  #$02            DATA REC'S ONLY
C3ED 26  D7            BNE   DISPLP
C3EF 8D  5B            BSR   DISPDA
C3F1 20  D3            BRA   DISPLP
                *
                *  DISPLAY HELP INFORMATION
                *
C3F3                HELPME
C3F3 BD  CD24          JSR   PCRLF
C3F6 CE  C60B          LDX   #HELPST
C3F9 BD  CD1E          JSR   PSTRNG
C3FC 20  C8            BRA   DISPLP
                *
                *  END OF FILE REACHED
                *
C3FE                ENDFIL
C3FE CE  C5B1          LDX   #EOFIL
                *
                *  PUT MSG, CLOSE FILES AND EXIT
                *
C401                ERRMSG
C401 BD  CD1E          JSR   PSTRNG
C404                CLOSE
C404 BD  D403          JSR   FMSCLS
C407 7E  CDD3          JMP   WARMS
                *
                **** ROUTINE TO SHOW RECORD ADDRESS
                *    AND SIZE
                *

C40A                DISPLY
C40A BD  CD24          JSR   PCRLF
C40D B6  C251          LDAA  TYPE
C410 81  16            CMPA  #$16
C412 27  25            BEQ   DISTRT
                *
                *  SHOW DATA RECORD
                *
C414 CE  C54C          LDX   #SRECAD
C417 BD  CD1E          JSR   PSTRNG
C41A CE  C252          LDX   #SADS
C41D BD  CD45          JSR   OUTADR
C420 86  2D            LDAA  #'-
C422 BD  CD18          JSR   PUTCHR
C425 CE  C104          LDX   #EADS
C428 BD  CD45          JSR   OUTADR
C42B CE  C55C          LDX   #SBYTEC
C42E BD  CD1E          JSR   PSTRNG
C431 CE  C254          LDX   #RECNT
C434 BD  CD3C          JSR   OUTHEX
C437 20  0C            BRA   ENDDIS
                *
                *  SHOW START RECORD
                *
C439                DISTRT
C439 CE  C569          LDX   #SSTART
C43C BD  CD1E          JSR   PSTRNG
C43F CE  C252          LDX   #SADS
C442 BD  CD45          JSR   OUTADR
                *
                *  PROMPT HUMAN
                *
C445                ENDDIS
C445 CE  C579          LDX   #PROMPT
C448 BD  CD1E          JSR   PSTRNG
C44B 39                RTS
```

```
                        *
                        ****DISPLAY DATA ROUTINE
                        *
C44C                    DISPDA
C44C BE  C252                   LDX   SADS
C44F BF  C109                   STX   TSADS
C452 BE  C255                   LDX   #DATBUF
C455 BF  C10D                   STX   BUFPNT
C458 B6  C254                   LDAA  BCNT
C45B B7  C10F                   STAA  TBCNT
                        *
                        *   SHOW ADDRESS THIS LINE
                        *
C45E                    OUTLUP
C45E BD  CD24                   JSR   PCRLF
C461 BE  C109                   LDX   #TSADS
C464 BD  CD45                   JSR   OUTADR
C467 B6  10                     LDAA  #16
C469 B7  C110                   STAA  LINC
C46C 1F  8943                   TAB
C46F BE  C109                   LDX   TSADS
C472 BD  CD36                   JSR   ADDBX
C475 BF  C109                   STX   TSADS
C478 BE  C10D                   LDX   BUFPNT
C47B BF  C10D                   STX   BPOINT
                        *
                        *   SHOW DATA AS HEX LOOP
                        *


C47E                    HEXLUP
C47E B6  20                     LDAA  #$20
C480 BD  CD18                   JSR   PUTCHR
C483 BD  CD3C                   JSR   OUTHEX
C486 30  01                     INX
C488 7A  C10F                   DEC   TBCNT
C48B 27  07                     BEQ   ENDREC
C48D 7A  C110                   DEC   LINC
C490 26  EC                     BNE   HEXLUP
C492 20  03                     BRA   PRTCHR
C494                    ENDREC
C494 7A  C110                   DEC   LINC
                        *
                        *   SHOW DATA AS ASCII
                        *
C497                    PRTCHR
C497 BF  C10D                   STX   BUFPNT
C49A BD  CD24                   JSR   PCRLF
C49D B6  10                     LDAA  #16
C49F B0  C110                   SUBA  LINC
C4A2 B7  C110                   STAA  LINC
C4A5 B6  7C                     LDAA  #$7C      VERTICAL BAR
C4A7 BD  CD18                   JSR   PUTCHR
C4AA BE  C10D                   LDX   BPOINT
C4AD                    CHRLUP
C4AD A6  84                     LDAA  0,X
C4AF 30  01                     INX
C4B1 84  7F                     ANDA  #$7F
C4B3 81  20                     CMPA  #$20
C4B5 25  04                     BLO   PRTPER    LESS THAN SPACE
C4B7 81  7F                     CMPA  #$7F
C4B9 26  02                     BNE   PRTCHR    OR RUBOUT
C4BB                    PRTPER
C4BB B6  2E                     LDAA  #'.
C4BD                    PRTCHR
C4BD BD  CD18                   JSR   PUTCHR
C4C0 7A  C110                   DEC   LINC
C4C3 26  E8                     BNE   CHRLUP
C4C5 B6  7C                     LDAA  #$7C      VERTICAL BAR
C4C7 BD  CD18                   JSR   PUTCHR
C4CA 7D  C10F                   TST   TBCNT
C4CD 26  BF                     BNE   OUTLUP
C4CF 39                         RTS
                        *
                        ****GET ROUTINES
                        *
                        *   GET RECORD ROUTINE
                        *
C4D0                    GETREC
C4D0 BD  29                     BSR   GETADS
C4D2 B6  C251                   LDAA  TYPE
```

```
C4D5 81  16                     CMPA  #$16
C4D7 27  2E                     BEQ   ENDETR
C4D9 BD  2B                     BSR   GETBYT
C4DB B7  C254                   STAA  BCNT
C4DE 4A                         DECA
C4DF 1F  8940                   TAB
C4E2 BE  C252                   LDX   SADS
C4E5 BD  CD36                   JSR   ADDBX
C4E8 BF  C104                   STX   EADS
C4EB BE  C255                   LDX   #DATBUF
C4EE F6  C254                   LDAB  BCNT
C4F1                    NXTBYT


C4F1 BD  13                     BSR   GETBYT
C4F3 A7  84                     STAA  0,X
C4F5 30  01                     INX
C4F7 5A                         DECB
C4F8 26  F7                     BNE   NXTBYT
C4FA                    ENDETR
C4FA 39                         RTS
                        *
                        *   GET ADDRESS ROUTINE
                        *
C4FB                    GETADS
C4FB BD  09                     BSR   GETBYT
C4FD B7  C252                   STAA  SADS
C500 BD  04                     BSR   GETBYT
C502 B7  C253                   STAA  SADS+1
C505 39                         RTS



                        *
                        *   GET BYTE ROUTINE
                        *
C506                    GETBYT
C506 BF  C107                   STX   TEMPX
C509 BE  C840                   LDX   #$FC0A
C50C BD  0406                   JSR   FMS
C50F 26  04                     BNE   ERREAD
C511 BE  C107                   LDX   TEMPX
C514 39                         RTS
C515                    ERREAD
C515 BE  C59A                   LDX   #SEREAD
C518 7E  C401                   JMP   ERRMSG
                        *
                        ****PUT ROUTINES
                        *
                        *   PUT RECORD ROUTINE
                        *
C518                    PUTREC
C518 BE  C251                   LDX   #TYPE
C51B C6  03                     LDAB  #$03
C51D BD  0E                     BSR   PUTBYS
C51F B6  C251                   LDAA  TYPE
C522 81  16                     CMPA  #$16
C524 27  06                     BEQ   EXDPTR
C526 F6  C254                   LDAB  BCNT
C529 5C                         INCB
C52A BD  01                     BSR   PUTBYS
C52C                    EXDPTR
C52C 39                         RTS
                        *
                        *   PUT BYTES ROUTINE
                        *
C52D                    PUTBYS
C52D A6  84                     LDAA  0,X
C52F 30  01                     INX
C531 BF  C107                   STX   TEMPX
C534 BE  C811                   LDX   #FC0OUT
C537 BD  0406                   JSR   FMS
C539 26  07                     BNE   EWRITE
C53B BE  C107                   LDX   TEMPX
C53E 5A                         DECB
C53F 26  EC                     BNE   PUTBYS
C541 39                         RTS
C542                    EWRITE
C542 BE  C5A5                   LDX   #SEWRIT
C545 7E  C401                   JMP   ERRMSG
```

```
                        *
                        ****STRINGS AND THINGS
                        *
C54C                    SREEAD
C54C 41 44 44 52                FCC     'ADDRESS RANGE: '
C550 45 53 53 20
C554 52 41 4E 47
C558 45 3A 20
C55B 04                         FCB     04
C55C                    SBYTEC
C55C 42 59 54 45                FCC     'BYTE COUNT: '
C560 20 43 4F 55
C564 4E 54 3A 20
C568 04                         FCB     04
C569                    SSTART
C569 53 54 41 52                FCC     'START ADDRESS: '
C56D 54 20 41 44
C571 44 52 45 53
C575 53 3A 20
C578 04                         FCB     04
C579                    PROMPT
C579 45 4E 54 45                FCC     'ENTER COMMAND (C,W,B,E,H=HELP): '
C57D 52 20 43 4F
C581 4D 4D 41 4E
C585 44 20 28 43
C589 2C 4E 2C 44
C58D 2C 45 2C 48
C591 3D 48 45 4C
C595 50 29 3A 20
C599 04                         FCB     04
C59A                    SEREAD
C59A 52 45 41 44                FCC     'READ ERROR'
C59E 20 45 52 52
C5A2 4F 52
C5A4 04                         FCB     04
C5A5                    SEWRIT
C5A5 57 52 49 54                FCC     'WRITE ERROR'
C5A9 45 20 45 52
C5AD 52 4F 52
C5B0 04                         FCB     04
C5B1                    SEOFIL
C5B1 45 4E 44 20                FCC     'END OF FILE ENCOUNTERED'
C5B5 4F 46 20 46
C5B9 49 4C 45 20
C5BD 45 4E 43 4F
C5C1 55 4E 54 45
C5C5 52 45 44
C5C8 04                         FCB     04
C5C9                    EINPUT
C5C9 43 4F 4D 4D                FCC     'COMMAND FORMAT ERROR'
C5CD 41 4E 44 20
C5D1 46 4F 52 4D
C5D5 41 54 20 45
C5D9 52 52 4F 52
C5DD 04                         FCB     04
C5DE                    EOPWRD
C5DE 49 4E 50 55                FCC     'INPUT FILE OPEN ERROR'
C5E2 54 20 46 49
C5E6 4C 45 20 4F
C5EA 50 45 4E 20
C5EE 45 52 52 4F
```

TO BE CONTINUED

BIT BUCKET

Karl Lunt
2119 W. Cholla
Phoenix, AZ 85029

I have used a Gemini 10 printer on my homebrew 6809 system for some time now and have always been somewhat discouraged about FLEX's inability to store escape codes in text files. This inability prevented me from imbedding font and graphics commands within a text file, for altering the print characteristics of the Gemini 10 at print time.

Not too long ago I spent some time helping a friend get his Epson printer running on his Apple and was really impressed with the fancy printing he was able to do with imbedded print control sequences. The fact that an Apple with its primitive DOS could do something that my 6809 with FLEX couldn't do finally got to me and I decided to create the badly needed software.

The resulting program is a more powerful version of PRINT.SYS than is supplied in the FLEX user manual. The first thing to note is that due to the enhancements, the new PRINT.SYS will no longer fit in the hole left by FLEX at $ccc0-$ccfb. As a result, only the entry points to the needed printer functions are left in their proper place and the actual character output code is moved to another location in the memory map. I happen to have a hole in the area above $e200, so I moved my output code to $e300. The output code is completely position independent and may be moved anywhere you have room. Obviously you will have to edit the ORG statment at PROUT to reflect any change in the code's location.

The new PRINT.SYS routine will behave normally for all characters sent to the line printer until a backslash is detected. When this happens, a flag is set to indicate that a backslash was encountered and the backslash itself is ignored. The character following a backslash is checked to see if it is in the range $40 through $5F. If it is (with one exception), it is turned into a control character and sent to the printer in place of the backslash-character combination.

The single exception to the above rule is a backslash-backslash combination. I chose to interpret this as a single backslash. This means that my PRINT.SYS routine will not send a control-backslash ($1C or the ASCII FS character). If you feel differently about this, feel free to alter the code accordingly.

With the new PRINT.SYS routine installed, it is a simple matter to send any control or escape sequences to the printer. For example, to send a BEL character, just insert a backslash-G in your text file. Similarly, the escape code is a backslash-[.

Note that these backslash codes can appear in any type of file that might be sent to a printer. Thus, it is easy to put fancy titles on your assembler listings, by inserting the appropriate backslash codes in the title lines and in the comments. With a little imagination, you can develop graphics additions to your programs and listings.

The source file is assmbled with the standard FLEX assembler instruction:

```
(P) ASMB PRINT.TXT (+Y)
```

and the output binary file should be transferred to your system disc with the name PRINT.SYS. Reboot your DOS and execute a printer command (such as P DIR) and the new printer driver will be loaded into memory. Subsequent listing of a file with backslash codes in it should produce the appropriate printer characteristics changes. For example, sending a backslash-[-4 to the Gemini 10 should cause the printer to output text in italic mode.

- - -

```
*
* PRINT.SYS DRIVER FOR SERIAL PRINTER.  TAKEN FROM
* FLEX MANUAL, PAGE 3.9.  MODIFIED FOR USE WITH A
* SERIAL PORT AT $E00C.
*
* MODIFIED 17JUL84 TO PERMIT TRANSMISSION OF ESC
```

```
                    * SEQUENCES TO THE LINE PRINTER.  ESC (AND ANY
                    * OTHER CONTROL CHARACTER) IS SIGNALED BY A
                    * BACKSLASH.  THE CHARACTER FOLLOWING THE BACKSLASH
                    * (PROVIDED IT IS IN THE RANGE $40 TO $5F) IS
                    * CONVERTED TO A CONTROL CHARACTER AND SENT IN
                    * PLACE OF THE BACKSLASH-CHAR SEQUENCE.
                    *

           E00C ACIA    EQU   $E00C     ACIA ADDRESS OF PRINTER

                    *
                    * PRINTER INITIALIZATION.
                    *
CCC0                     ORG   $CCC0
CCC0 86  13      PINIT   LDA   #$13     RESET ACIA
CCC2 B7  E00C            STA   ACIA
CCC5 86  11              LDA   #$11     SET 8 BITS & 2 STOP BITS
CCC7 B7  E00C            STA   ACIA
CCCA 39                  RTS            RETURN

                    *
                    * CHECK FOR PRINTER READY.
                    *
CCD0                     ORG   $CCD0
CCD0 34  04      PCHK    PSHS  B        SAVE BR
CCD2 F6  E00C            LDB   ACIA     GET STATUS
CCD5 56                  RORB           MOVE TDR BIT INTO
CCD6 56                  RORB           SIGN POSITION
CCD7 56                  RORB
CCD8 35  04              PULS  B        RESTORE BR
CCDA 39                  RTS            RETURN

                    *
                    * PRINTER OUTPUT CHAR ROUTINE.
                    *
CCE4                     ORG   $CCE4
CCE4 7E  E300    POUT    JMP   PDOUT    TRANSFER TO THE OUTPUT AREA
CCE7 00          BSFLAG  FCB   0        BACKSLASH FLAG STORED HERE

                    *
                    * NEW OUTPUT SECTION.  MOVED HERE BECAUSE FLEX
                    * DOESN'T LEAVE ENOUGH ROOM IN THE SYS AREA.
                    *
                    * NOTE THAT A BACKSLASH-BACKSLASH SEQUENCE WILL
                    * BE SENT AS A SINGLE BACKSLASH.  THIS MEANS THAT
                    * AS PRESENTLY IMPLEMENTED, IT IS NOT POSSIBLE
                    * TO SEND A CTRL-BACKSLASH.
                    *
E300                     ORG   $E300
E300 84  7F      PDOUT   ANDA  #$7F     CLEAR TOP BIT
E302 70  CCE7            TST   BSFLAG   BACKSLASH ACTIVE?
E305 27  2A              BEQ   PR1      BRANCH IF NOT
E307 7F  CCE7            CLR   BSFLAG   YES, CLEAR IT
E30A 81  5C              CMPA  #$5C     IS CHAR BACKSLASH?

E30C 27  04              BEQ   SEND     YES, SEND IT
E30E B5  20              BITA  #$20     CAN CHAR BE A CTRL-CHAR?
E310 26  15              BNE   PR2      BRANCH IF NOT
E312 B5  40              BITA  #$40     IS CHAR ALREADY A CTRL-CHAR?
E314 27  11              BEQ   PR2      BRANCH IF YES
E316 84  1F              ANDA  #$1F     NO, MAKE IT A CTRL-CHAR

                    *
                    * THIS SECTION OF CODE WILL OUTPUT THE CHAR IN
                    * THE AR TO THE LINE PRINTER.  AS REQUIRED BY
                    * FLEX, ALL REGISTERS ARE PRESERVED.
                    *
E318 34  04      SEND    PSHS  B        SAVE BR
E31A F6  E00C    SEND1   LDB   ACIA     GET STATUS
E31D 57                  ASRB           PUT TDR BIT IN CARRY
E31E 57                  ASRB
E31F 24  F9              BCC   SEND1    LOOP TIL DONE
E321 35  04              PULS  B        RESTORE BR
E323 B7  E00D            STA   ACIA+1   WRITE OUT CHAR
E326 39                  RTS            RETURN

                    *
                    * CONTROL REACHES THIS POINT IF A BS WAS FOLLOWED BY
                    * A CHARACTER THAT CANNOT BE MADE A CONTROL CHAR.
                    * IN THIS CASE, SEND A BACKSLASH FOLLOWED BY THE
                    * ACTUAL CHARACTER.
                    *
E327 34  02      PR2     PSHS  A        SAVE AREG
E329 86  5C              LDA   #$5C     GET A BACKSLASH
E32B BD  EB              BSR   SEND     AND SEND IT
E32D 35  02              PULS  A        RESTORE CHAR
E32F 20  E7              BRA   SEND     AND SEND IT

                    *
                    * CONTROL REACHES THIS POINT IF NO BACKSLASH WAS
                    * ACTIVE.  THIS IS THE NORMAL CHARACTER OUTPUT
                    * SEQUENCE.
                    *
E331 81  5C      PR1     CMPA  #$5C     IS CHAR A BACKSLASH?
E333 26  E1              BNE   SEND     NO, SEND NORMALLY
E335 7C  CCE7            INC   BSFLAG   YES, SHOW BACKSLASH
E338 39                  RTS            AND IGNORE CHAR

                         END

0 ERROR(S) DETECTED

SYMBOL TABLE:

ACIA   E00C   BSFLAG CCE7   PCHK   CCD0   PINIT  CCC0   POUT   CCE4
PR1    E331   PR2    E327   PDOUT  E300   SEND   E318   SEND1  E31A
```

```
1000 REM *** FILECOMP.BAS - compare 2 ascii files, esp. BASIC.
1010 REM      By L.P.L. Piacenza
1020 REM         Chemistry Dept., University of Transkei,
1030 REM         Private Bag 15092, UMTATA,
1040 REM         Republic of Transkei, Southern Africa.
1050    FA%=1 : FB%=2 : CL$=CHR$(12) : BL$=CHR$(7)
1100    OP$="old: " : NP$="new: "
1150    PRINT CL$
1200    PRINT "OLD PROGRAM NAME "; : INPUT LINE OL$
1250    I$=OL$ : GOSUB 4350 : OL$=I$
1300    PRINT "NEW PROGRAM NAME "; : INPUT LINE NW$
1350    I$=NW$ : GOSUB 4350 : NW$=I$
1400    IF OL$=NW$ THEN PRINT BL$;"CANNOT USE SAME FILENAME!" : END
1450    INPUT"WANT PRINTER ",PR$
1500    IF PR$(0)="Y" THEN 1600
1550    OPEN "O.PRINT" AS 0
1600    PRINT CL$
1650    PRINT #0,TAB(65);DATE$
1700    PRINT TAB(18);"Comparison printout of the files:"
1750    PRINT #0,TAB(20);"OLD PROGRAM - ";OL$
1800    PRINT #0,TAB(20);"NEW PROGRAM - ";NW$ : PRINT #0 : PRINT #0
1850    OPEN OLD OL$ AS FA% : OPEN OLD NW$ AS FB%
1900    PRINT : PRINT
1950    GOSUB 3650
2000    IF FA%=0 THEN 3900
2050    GOSUB 2800 : REM ***** READ OLD LINE
2100    IF FA%=0 THEN 3900
2150    IF FB%=0 THEN 4150
2200    GOSUB 3150 : REM ***** READ NEW LINE
2250    GOSUB 3650
2300    IF FA%=0 THEN 3900
2350    IF FB%=0 THEN 4150
2400    GOSUB 2500
2450    IF POI<PNI THEN 1950 ELSE 2250
2500    POI=VAL(LEFT$(A$,5))
2550    PNI=VAL(LEFT$(B$,5))
2600    IF POI<PNI THEN GOSUB 3450 : RETURN
2650    IF POI>PNI THEN GOSUB 3750 : GOSUB 2800 : RETURN
2700    GOSUB 3800 : GOSUB 3150 : RETURN : REM ***** POI>PNI
2750 REM ******* read one line from OLD file.
2800    ON ERROR GOTO 3000
2850    IF FA%=0 THEN RETURN
2900    INPUT LINE #FA%,A$
2950    RETURN
3000    IF ERR=8 THEN CLOSE FA% : FA%=0 : RESUME 2950
3050    ON ERROR GOTO
3100 REM ******* read one line from NEW file.
3150    ON ERROR GOTO 3350
3200    IF FB%=0 THEN RETURN
3250    INPUT LINE #FB%,B$
```

```
3300    RETURN
3350    IF ERR=8 THEN CLOSE FB1 : FB1=0 : RESUME 3300
3400    ON ERROR GOTO
3450 REM **** compare 2 lines, of same number, for equality.
3500    IF A$<>B$ THEN PRINT #0,CP$;A$ : PRINT #0,MP$;B$ : PRINT #0
3550    RETURN
3600 REM *** check if both files closed = done!
3650    IF FA1=0 AND FB1=0 THEN 4550 ELSE RETURN
3700 REM *** print line headers.
3750    PRINT #0,OP$;A$ : PRINT #0 : RETURN
3800    PRINT #0,MP$;B$ : PRINT #0 : RETURN
3850 REM **** case where old file has ended.
3900    GOSUB 3150
3950    IF FB1=0 THEN 4550
4000    GOSUB 3800
4050    GOTO 3900
4100 REM **** case where new file has ended.
4150    GOSUB 2800
4200    IF FA1=0 THEN 4550
4250    GOSUB 3750
4300    GOTO 4150
4350 REM ***** EXTENSION
4400    IF LEFT$(RIGHT$(I$,0),1)="." THEN RETURN
4450    I$=I$+".BAS"
4500    RETURN
4550    IF PR$="Y" THEN CLOSE 0
4600    END
```

Notes on upgrading to 2 Mhz & Extended Addressing

While upgrading to 2 Mhz and extended addressing I noticed a few things. With SWTP FLEX and Extended addressing FLEX sets up the MAT to use 0xxxx for it self and 1xxxx for program memory. If you abort a program by hitting reset memory will be set to all 0xxxx by SBUG-E. So any program in memory is lost. it's in extended memory somewere. FLEX also acts funny if you jump back to it. The only thing to do is Re-Boot. I added a DC-4 double density disk controller to my system and had to switch the I/O to slow or the disk controller would not work. I looked into just how much it was being slowed by and found that for a 2 Mhz clock the I/O was running at approx. 815 Khz - Why so slow ?. What I did was to change the value of C1 from 220 pf to 56 pf to give me an I/O speed of approx. 1.5 Mhz, I experimented to find this value you may have to do the same. Try a 56 pf or a 68 pf to start with this should put you in the ball park. Over a period of time my startup routine was growing and taking longer. I have found a way to speed up some of it by putting my TTYSET and ASN values into the FLEX core. This can be done with the FIX utility to change FLEX's default values. Once this is done TTYSET and ASN commands can be deleted from your startup routine. The same approch can be used with the SBOX utility if you have to set values at startup. The best way to find out what value to set SCC33 of the SBOX command byte is to set it with SBOX goto MON and look what is at that location. With this done my startup routine has only to load my clock/timer. I'm sure you will benefit by decreasing boot time.

Joseph M Aulicino

Joseph M. Aulicino
2014-59th Street
Bklyn, N.Y. 11204

USING LSET AND RSET WITH REGULAR STRINGS

The manual for TSC's BASIC says that the LSET and RSET statements may be used with regular strings but it doesn't show you how.

The following program is to demonstrate the use of LSET and RSET with regular strings:-

```
10 REM USING LSET AND RSET
20 REM WITH REGULAR STRINGS
30  A$="1234567890"
40 INPUT"ENTER STRING";X$
50  LSET A$=X$
60 PRINT A$
70  RSET A$=X$
80 PRINT A$
90 PRINT
100 GOTO 40
```

Line 30 sets up the string storage space for 10 characters. The string entered by the user in line 40 will be printed out left justified [lines 50 & 60] and then right justified [lines 70 & 80]. The string storage space can be varied by changing the length of A$.

C.S.Soh
67-E Blk3 Marine Vista
Singapore,1544
Republic of SINGAPORE.

Ed Cole & Associates
2806 Short Ridge
San Antonio, TX 78231
(512) 340-3957 - 492-6071

Dear Don:

During the next three months Ed Colle and Associates will have extra time to devote to consulting and equipment design.

Ed Colle and Associates has been a Consultant since 1974 and is registered with Motorola's Consultant Support Program as a Consultant familiar with Motorola's microprocessors.

We also have worked as a Consultant with Texas Instruments on the design of custom chips for a customer involved with the mass consumer market, where production of in excess of 1 million units per year is not uncommon.

The Industrial Designers and Plastics Designers that work with us on the human factors aspects and mold designs are the very best available in the San Antonio area.

My experience in electronics covers a period of 35 years. The first twenty as a field engineer and the last fifteen as an equipment designer. During the early part of my career I worked on the leading edge high tech (military) prototypes of the period.

As an electronics designer beginning at the very start of the low cost computer explosion, I was involved in the design of many of the pieces of equipment that added fuel to the fire to keep it going. Several of the very first under $2000 terminals were my designs, vintage 1969 to 1972. The SWTPC CT-1024 also known as the TV-TYPEWRITER 2 was my design, vintage 1973.

Some of the SS-50 bus advances were either designed by me or based on prototypes built and designed by me. One good example is the Intelligent Marksman Controller sold as the CDS-1 (vintage 1977) which contained design concepts just now appearing in the latest equipment, and was the very first commercial use of DMA in personal computer equipment.

Please let me clarify this last statement. The

design was completed and sold in 1977, although production was not started until 1979. By this time the company had introduced other products using DMA. Other SS-50 bus designs were 9600 baud tape units (1976), I/O processor (1979), 32K static ROM - Ram board (1977), extremely high speed intelligent swapping disk(1980), CSMA data transfer link (uniflex 1980).

In addition, some of the advances such as memory management, multi-layer boards, intelligent peripherals, PLL's, PAL'S and multi-ported memories were first seen by SS-50 buss designers in designs by Ed Colle and Associates for other clients.

Ed Colle and Associates also does packaging and detailing. Some of the business computers, in use today, were packaged, and the mechanical structures designed by our group. A popular line of computer furniture was designed by our group (1972-1975). We have in the past done mechanical design of printer mechanisms, cut sheet feeders, envelope feeders and even designed the mechanical structure of digital tape recorders (1971 and still in production). Key switches and keyboards for one of the earliest low cost business computer, and terminal manufacturers were designed and detailed by our group.

We have designed cases and built mold models for several very successful consumer product lines, as well as for a very popular family of computers.

In addition to the above we have also, set up manufacturing facilities for computers, home products, advertising signs, and insecticide sprayers.

We designed and installed an advertising network of 21 large bill-board type electronic message signs. These signs are electronicly programmed from a central location, over local phone lines and capable of operating unattended, from stored programs. If service was needed the signs could report back to the central office.

Many of the large advertising signs and sign programmers designed and built by our group in 1973 are still in service and have during the intervening 11 years required very little service. These signs were 300 kilowatt monsters with in excess of 1000 incandescent light bulbs. The energy crisis in 1974, ended the manufacture of the monster signs, but several hundred of the smaller signs have been built from our designs (10-100 kilowatts).

A line of gas station pricing signs was designed and manufactured in 1973 and again in 1980. The early design was for a sign company and the later for an oil company. In both cases the first 100 units were built by our group as a prototype run.

The appearance of the product has a lot to do with the acceptance of the product by the buying public and in many cases the appearance and reliability of the product are more important than the performance. We try to balance the appearance, reliability and performance concepts to create a successful product.

Several of the latest projects involve custom integrated circuits. A beer inventory control system currently being used in restaurants, race tracks and stadiums uses a 6805. An electronic room air freshener uses HCMOS Standard Cell. A chemical mixing station currently being designed will use a 6801. An electronicly controlled insect fogger designed by us in 1977, will be redesigned using Programed Logic Arrays.

In addition to the design work, we also found time to get involved with data processing. For three years we handled all of the transactions for several of the largest Barter banks in this area, providing daily printout of customer accounts.

We have capabilities in most fields of electronics and if we have not already done it, we probably could do it. If you have customers who went us to support them in their projects please do not hesitate to refer them to us.

Ed Colle

[illegible] Engineering Co.

91 S. Leigh Ave., Campbell, CA 95008

September 19 1984 (408) 371-4573

6809 users!

For several years I have been using the 6809 and related Products, and am very tired of being pushed into the dark ages by other non-6809 Products. It seems to me that the time is right for a nice portable.

Tired of spending endless hours writing software for a machine that "they" now say is obsolete junk. The market all went the other way", so now you might as well scrap "that" stuff and buy "our" new model. I for one want off the treadmill, and the only way to is for "us" the users have control over the product.

We need a good 6809 based portable that can have a useful life of 5 years or so. There will always be more, but for many uses the configuration described later will be quite adequate.

It seems that so much of the problem is that "they" don't want to preserve OUR investment, but rather obsolete it as fast as possible. (true/still or genius I'd say). Useful life is largely determined by the size and hence clout of the user base. Apple II's are still quite useful (if you like apples) because of the large user base and because they have kept making essentially the same product for such a long time.

The 6809 has, in my opinion, suffered greatly at the hands of Moto and the Radio boys. I got the word that the 68EC09 is on the block due to lack of "interest", and unless that gets turned around, we may be further relegated to the dark ages.

Now to the real point. For several years (maybe 3) I have felt that a 6809 based portable product was very much wanted and needed. A nice unit that could fit in a brief case. Since I don't have any money or clout, I wanted to turn to you, the 6809 community, for a hearing. If we could a we have bee together and form a company that was primarily owned and run by 6809 use a, this and many other currently impossible dreams could come true.

The main features of the machine in mind are as follows:

- *) 68309 processor
- *) 80 X 25 line LCD flat dis lay w/ graphics ability.
- *) 3.5" floppy disk drive and controller with one drive inside, 3 drives allowed outside (can be 5.25")
- *) 712A comparable codes built in, w/ firmware terminal emulator.
- *) 56k CMOS ram
- *) 8k of Eram/Eprom/Rom/Exe Monitor etc.
- *) 256 bytes I/O space
- *) Parallel printer port (centronics)
- *) External bus access for I/O, board growth, etc.
- *) All hardware and firmware source provided with signed license.

If this sounds good to you, please write (don't call as I can't show that to investor types) and give me your thoughts. Would this kind of machine be worth around $1.3K - $2.0k especially if it were well supported by the 6809 user community ?

It still takes a lot of good software to make this kind of thing fly. This configuration should be able to support much of the stuff that's now out there (Flex9, and OS9). It would also open lots of new opportunities for users as well as system devel pers.

Thanks for taking the time to read this.
Please write today !

Robert Nixon
Nixon Engineering Co.

Tom Gilchrist
1450 N. Clarence ,108
Wichita, KS. 67203

MODEM9+
UPDATES TO MODEM9

When Norm Commo published the MODEM9 program in the April 1984 issue of 68 Micro Journal, I was looking for a modem program for FLEX in 'C'. I had always wanted a modem program for FLEX with the XMODEM protocol (Ward Christianson's fine standard). I wanted a program I could mess with, and since I don't work in assembler (as AT+T said in a question about assemblers on their 3B, "You know, this is 1984."), Norm's program was just what I wanted.

Anyway, I got the source entered and compiled it using INTROL's compiler (now version 1.5+). With some help from Brad Taylor, we added a number of features to the fine original program. The program is used to transfer text and binary over the phone between a number of FLEX users here in town and I use it to communicate with CP/M and UNIX systems (more on UNIX later).

This program has only been compiled with INTROL's compiler, but it should work with any standard FLEX box. I use a Peripheral Technology computer with 80 track 5 1/4" drives. The listing is the version I am now using and is set up the same as the original.

MODEM.H	Headers
MODEM1.C	main()
MODEM2.C	xmodem work
MODEM3.C	file i/o

The header file includes the defines for the addresses of the ports. The code is set up for the following

hardware:

```
   Terminal    $E004
   Modem       $E000 (Set p  t for 1200 baud. Program
will set
                  to 300 baud on command "b" by setting
ACIA
                  divide rate.)
   Printer     $E070 PIA
```

You can configure the modem ACIA's by changing the defines in MODEM.H. You will note that in the MODEM1.C code, in the function "terminal()", I set the terminal ACIA when going into terminal mode and again when returning to the command mode. This code should be taken out if you are NOT running an interrupt driven driver for FLEX. I use a version of the TERM program published in the July 1983 issue of 68 Micro Journal and this code is needed to turn off keyboard interrupts. The standard MODEM9+ program, like the original, does not initialize the terminal ACIA anywhere in the program except the code mentioned above.

To compile, u e the following EXEC script (or type in from the terminal)...

```
   ICC MODEM1.C
   ICC MODEM2.C
   ICC MODEM3.C
   ILINK MODEM1 MODEM2 MODEM3 -T=5
```

The resulting program will be called MODEM1.CMD and can be changed to any name you desire. I have no idea if this code will compile with other 'C' compilers on FLEX. The code uses a "longjmp" for the ESC pr essing. This could cause a proble with some 'C' compilers.

I will not go into the operation of the program because it was well explained in the original column. However, I will give the changes and how they work.

Auto Dial

The auto dial feature will read an ASCII script from the file 0.MODEM.DAT when the program starts. When in the command ode and you type "a", the data base will be listed (up to 15 services can be stored and listed). This file is created and maintained with a standard editor and an example is shown below...

```
Compuserve
1200
6557895
68 Micro Journal
300
16158426809
Dow/Jones News
1200
3450994
```

Notice that each phone entry is three lines. The first line is the name of the service (up to 35 characters), the baud rate (300 or 1200), and the last line is the phone number to be dialed (up to 18 characters).

You simply enter the number listed beside each service on the screen, then you are given the phone number and the baud rate. If you want to continue, you type "y" and the program will send the information out to the modem, the number will be dialed, and you will be put in the terminal mode. You can type a "CR" at any time during the data base work and you will return back to the command mode. To get out of the terminal mode you type a "CTRL @".

Commands:

I have added a number of commands other than the "a" listed above. I have changed the "u" upload and "d" download to "s" send and "r" receive. I always get confused about the terms upload and download. I always have to take a moment to think if I am uploading a file from my computer to the host or if I am uploading code from the host. Then I have the problem of figuring out who is the host. As you can see, I am easily confused and send and receive helps me. If you make a mistake and enter an "s" or "r", simply enter a CR for the file name and you will return back to the command mode.

I have changed the "s" command (give the terminal settings) to "I" for information and a CR, while in the command mode, will also bring the info up.

The "J" command toggles the Dow/Jones mode which strips off any $1E when receiving text in the terminal mode. I added this because the Dow/Jones service uses the $1E on each frame and my terminal homes the screen without clearing it. This makes for a confusing session.

In the old days, half duplex was used on dial up lines. This is 1984 and very few teletypes are still in use, so most modern computers support full duplex. However, no one ever told IBM about this, so on their big mainframes (like the ones I use) you still need half duplex (or spend a lot of money for a front end dial-up computer). Anyway, the "m" command puts you in half duplex and will allow you to use the CTRL H key to erase characters (standard backspace on most all computers except for IBM/PC's...they were never told).

The original version of MODEM9 did no ESC processing; it was really not needed. But I sometimes hit the ESC key while in the command mode. If I type a CR, the progra will drop back to FLEX. Brad added the code for the ESC processing and now it works as it should (a CR will bring you back to MODEM9+).

Brad also added some code for the "x" command. This will allow you to enter a FLEX command (one that runs up in $C100). When the FLEX command is done, you will return to the command mode. I must say that as old as FLEX is, it stands up very well against the new OS's of today and the command space is a nice feature. However, it is safe to say that TSC did not fully think this one out (they were still in Indiana at the time and we all know about the mid-west). If you run a command that does not reside at $C100, bye bye. If you run a program that resides at $C100, but uses memory that overwrites the MODEM9+ code, bye bye. There are other problems, but I don't think you will run into them unless you have an interrupt driven terminal driver on your FLEX (like TERM on mine).

There are some ways around this problem. For instance, you could mark FLEX commands that will run safely at $C100 by writing a FLEX command that would set an unused byte in the target commands FCB. You would then check the byte before passing control to FLEX in MODEM9+. However, I did not feel like doing this and I wish upon a star, that if anyone ever re-writes FLEX, they will try to address the problem.

Transfers:

There are times when you might not be able to use XMODEM protocol in transferring a text file. The capture buffer can be used to receive and save a file. I have sized the buffer to about 24K. I have also added code to send an XOFF to the host if the buffer runs out (a bell sounds at the terminal). You could then save the file with a "k" (with a name like FILE1.TXT), then go back to the terminal mode and send an XON (CTRL Q). Then next buffer could be saved with the file name FILE2.TXT and so on. You can then use the FLEX APPEND command to append the sections together (not from the MODEM9+ program with the "x" command!). The code might miss one character when the XOFF is sent (I always leave at least one problem in a program...keeps me humble).

To send a file without XMODEM protocol, I have added some code to ask if you want to send a file with XMODEM or ASCII when using the send command ("s"). There is no XON-XOFF in this section. This would make a fine feature to add if you need it. I have never had problems, though I hardly ever use this feature.

Let me tell you about CP/M. Receiving a file from a CP/M computer (with the CP/M file mode set using XMODEM protocol) is a real adventure. Until you look at the transferred files on your FLEX disk, you can't be sure that you have a usable transfer. The files are transferred "as-is" in the original MODEM9 which is OK for some CP/M's. However the extra $0A's or $00's can cause extra work with an editor. I added code to strip the code before it gets to the disk. FLEX also has a nifty disk space compression for text files that uses a $09 (TAB) with a count. UNIX and CP/M use the $09 without a count, so when the files are written as-is (binary mode) to disk, the files read with tab spaces you will not believe! I added code to expand the tabs before they hit the disk. This works with the CP/M and UNIX systems I work with, the original MODEM9 code might work with other systems. Also, I have never tried to transfer a binary CP/M file. I shutter to think what would happen.

Another problem comes up with what are called "squeezed" files on CP/M systems. On some CP/M systems you can't decompress them before transfer when using the XMODEM protocol. This makes some sense because one of the reasons for squeezed files is to shorten the transfer time. The idea is to unsqueeze the files once on your

computer. What is needed is a FLEX squeeze and unsqueeze utility. There is source for these programs floating around and I am sure that they could be made to work on FLEX.

Other items:

I use UNIX at work and one of the reasons I use MODEM9+ is to transfer text to and from these computers. They are on dial-ups and I have installed a modem shell type program called UC. This program will allow you to call a UNIX system and send or receive text files using the XMODEM protocol (using the CP/M file mode of MODEM9+). The program is in 'C' and is in the public domain. The best version of the program is available in the VAX SIG on Compuserve. I have installed it with very little problem on a number of UNIX systems. If you try this, watch out for the "stty -cbreak" and for LAN's. XMODEM protocol on a LAN will probably need some special LAN channel set-up because of control code layer conflicts. Also, XMODEM protocol will not work half duplex on IBM mainframes using AMDAHL UTS UNIX.

I thought, wouldn't it be nice to have a FLEX computer on line during the day so I can call and send and receive files using XMODEM protocol? Why, it would almost be like having a BBS. I could have log-ins, a message of the day, and etc., just like the "big boys" running CP/M! I have just such a system using a version of the MODEM9+ program along with a few other programs to control security, XON/XOFF, type ahead, etc. I would have sent the article with this one, but I am thinking I can get a second free one year subscription from Don Williams if I send it in a few weeks from now!

Thanks to all those who did such a fine job on the original MODEM9 and to Brad Taylor for his contribution to this revised addition.

- - -

Don's Note: O.K. - right on, I and thousands of readers appreciate your input. Looking forward to the program (hinted?) about above. Thanks fellows, you and others like you who share with us all, makes it **ALL WORTH WHILE, THANKS!**

DMW

- - -

Editor's Note: Because of the length of the source files in C, and that they have been partially published in 68 Micro Journal previously (C User Notes), the updated source code is available on a 68 Micro Journal service disk. The disk will carry number 15. See notice elsewhere in this issue concerning reader service disk.

DMW

- - -

Dr L.P.L. Placen a
Chemistry Dept.
University of Transkei
Private Bag X5092
Umtata
Republic of Transkei
Southern Africa

September 10, 1984.
Mr Don Williams,
68 Micro Journal,
5900 Cassandra Smith,
P.O. Box 849,
Hixson, TN 37343,
USA.

Dear Sir,

I would like to express my opinion about something that has been worrying me for a while. While the S-100 bus community has a wide selection of computers to choose from, the 6809 fraternity has had little choice: either spend $5000 and up for a really professional system; or buy a 'toy' in the under $1000 class.

What I would really like, as a programmer and not a hardware hacker who could probably construct his own, is a machine with the following characteristics:

1. MC6809 running at 2 MHz.

2. Fully socketted, 56-64K, expendable to 256K or more, so that large files can be loaded, or a RAM disk used.

3. Be able to use more than 64K per application, as the 8-bit CP/M users are able to do. Why not have a $tt.to file larger than 64K? Why not have more than 48K for BASIC?

4. Expansion slot(s) so that commercially available SS50 cards can be used, whether memory or other goodies such as graphics cards.

5. Use either a terminal or a keyboard, 80 columns minimum. In the keyboard option the screen memory should be separate from the user RAM, but accessible for 'peeks' and 'pokes', and of 'high resolution'.

6. At least enough parallel and serial ports to take the usual peripherals, and leave a port or two for extras.

7. To be able to run any mix of drives (OSDD).

8. Could be supplied without RAM IC's to keep costs down, and let the purchaser buy as much RAM as he wants.

9. Run Flex and OS9 at will.

10. Supply all the drivers required.

11. Last, but not least, to cost well under $1000 (just the computer and necessary drivers, without peripherals)!

So far the only single-board machine I have seen advertised in 68 Micro, that has some of the above requirements, is the one from Artisan. How about some reviews on the Artisan, Peripheral Technology's PT69, Chandler Microsystems' CMI, DRC's *Unibaard*, and any others out there, apart from the established 6809 mainframes?

I am looking forward to other readers' views on this subject. A response from manufacturers would be most welcome.

Have you ever carried out, or considered carrying out, a survey of your readers' computer equipment for general interest and also to allow readers with similar equipment to communicate each other's problems and interests? I am thinking along the lines as shown:

UNIT	EXAMPLE (In my case).
COMPUTER	Gimix
CPU TYPE	6809 (1 or 2 MHz)
TERMINAL(s)/VIDEO BOARD(s)	SWTPc 8212
MEMORY	56K static (2x32K boards)
DOS	LEX (3)
(3 most used, 1 least used)	OS9 Level I (1)
PRINTER(s)	C. Itoh 8510
OTHER BOARDS/GOODIES	MP-R eprom programmer
	MP-N calculator board
	SWTPc AC-30 cassette interface

Finally, how about publishing a photograph of the 68 Micro Staff? I like to put faces to 'household' names.

Thank you for an informative magazine.

Yours sincerely,

Dr. L. Placenza.

- - -

Editor's note: Thanks for the letter(s) and programs. You folks in South Africa have sure been letting me know you are there and getting things done.

By the time this is published our series of reviews of just what you asked for (as well as hundreds of others) will have started. Seems this could be a big item and a new breath of life to our community of users. Thanks again - Keep in touch and best of luck.

DMW

- - -

P.S. As to the photo, I assume most readers would rather have something of value, a program or something instead. We try hard to 'waste' no space.

Don

September 1, 1984.

Computer Publishing Center
'68 Micro Journal
5900 Cassandra Smith
P.O. Box 849
Hixson, TN 37343
U.S.A.

Dear Don,

Working on a program lately, I discovered a possible hazard in the use of the FLEX 'DOCMND' ($CD4B) routine. When the routine is used, the address $CD91 will ALWAYS, independent of the actual stackpointer, be pushed onto the top of the standard FLEX stack i.e. $C07D/7E (for some reason it is not initialized to $C080). This is caused by the fact that all commands end with a JMP $CD03, consequently the following piece of code is executed:

```
CD03   JMP $CD6B
CD6B   LDS #$C07F   set up stack
  !
  !
  !    set up of interrupt etc.
  !
  !
CDBF   BSR $CDEB   first subroutine call
CD91  <----------  return address
```

Before the above subroutine reaches it's return, FLEX recognizes the DOCMND flag to be set and returns to the calling user program, but the return address remains on the FLEX stack.

Now, if you have created some routine that does not re-allocate the stack, and if you push a parameter on the stack (nearly all compilers do it that way) before you call a subroutine utilizing DOCMND, your program will probably crash, because the proper return address has been overwritten. It all depends on the number of bytes pushed onto the stack, you may be lucky just to return to FLEX in sort of a ware-start, but I think users should be aware of the danger inasmuch it is not documented in the FLEX manuals.

Sincerely,

Niels Oesten
Brostykkevej 189
DK-2650 Hvidovre
Denmark

Blue Jean Computer Engineering
2609 Choctaw Trail
Austin, Texas 78745
(512) 892-0494
August 28, 1984

Dear Editor,

With respect to the CRC hardware which I designed for GIMIX (see: "Cyclic Redundancy Check" by Mike Magnus, in October's '68 Micro Journal, pp. 33 - 34), I should like to correct the impression given in the article that the hardware which I designed is "non-sharable". That is incorrect.

The CRC hardware which I designed is sharable. In fact, it is the *only* device that I know of which is: the device is actually re-entrant. This was accomplished by making the internal CRC state available to the processor. The device can be used by an interrupting process at any time if the internal CRC state is first saved to memory. When the interrupting process is done, the saved CRC state can be recovered and placed back into the CRC device, which will then take up exactly where it left off.

The article is apparently saying that the user cannot expect to *see* such re-entrancy, which must thus be inhibited by some other part of the system. For example, if the CRC hardware is being fed from a DMA device, *that* device must *also* be re-entrant for the CRC process to be re-entrant.

The design for GIMIX was a hardware implementation of the 24-bit CRC polynomial which I created at Motorola, and was functionally equivalent to the (formerly) fast CRC routine which I wrote to implement the CRC for OS9. The prototype hardware was capable of computing a complete byte-CRC in every bus cycle, and was indeed "sharable".

Cordially,

Terry Ritter, P.E.

Classified Advertising

SWTPC Model **6212** Demonstator **Terminals** $695.
Tom (615)842-4600 Mon.-Fri. 10-5 EST.

TELETYPE Model 43 PRINTER - with serial (RS232) interface, and full ASCII keyboard. **LIKE NEW** - New cost $1295.00 - ONLY **$759.00** ready to run - Call Tom - Larry - Bob, CPI 615 842-4600

***

SWTPC 6809 Computer, 56K Memory, Serial/Parallel/Tape Interfaces, two Floppy Controllers, two EPROM Burners, Extender Card, $825.
Bill (312) 824-2317

***

6809 System with 6809 CPU, 64K Dram Card, Two Serial I/O Cards, F&D Assoc. Disk Interface, 12 Slot Motherboard with 8 I/O Slots. Also Included are Two 40 Track Floppy Drives with Case and Power Supply and Heath H19 Terminal. System is fully operational. Also Included is FLEX9 OS with 2 Basics and Screeneditor Package. Other Software goodies Included. $1200/BO
Call Jim @ (201)299-9499 or write PO Box 497C, Convent Station, NJ 67961.

***

Seals 8K Memory Bd. $50, DRC 64K Memory Board with 40K $120, DDC-16 Floppy Disk Controller $100, Shuggart SA-400 $75, MP-R 2716 Programmer $40, CT-64 Terminal $70, Visual 50 Terminal $475. Documentation for all included. Kurt Wolfe 5931 W. 41st Place Indpls., IN. 46254 (317)293-9701 (nights) (317)240-2752 (days).

***

Help

Dear Sir,

We are a group of people working with the 6809 microprocessor of Motorola and FLEX operating system.

Some of us have shown interest for the APL language.

Please, let me know the possibility to get this software under the FLEX 09, with all the comments to adapt it to hardware (keyboard from maxiswitch EJE-78).

How much does it cost?

Yours sincerely,

Jean-Yves COUSIN
34, route de Compiegne
02600 Villers-Cotterets
FRANCE

K-BASIC
for OS9 & FLEX
$199

K-BASIC is a complete BASIC compiler package including: the compiler itself; the assembler; documentation; and sample programs. It features six atomic data types including: real numbers, strings; 8 bit, 16 bit, 32 bit, and 64 bit signed integers. All types may be dimensioned with one or two subscripts. K-BASIC converts programs to MACHINE language code which may be put into EPROMS or ROMS.

K-BASIC syntax is very close to TSC's BASIC and XBASIC interpreters. Line numbers are not required (may be up to 16 characters). Variable names may be up to 12 characters long. The AT statement dimensions variables to absolute memory addresses.

The future of K-BASIC will see additional versions for the assorted interpreters currently available. This means you can compile your BASIC programs you now have.

Call (503) 666-1097 for our CATALOG. we have many other programs including: DO ... $69 OSM ... $99 ED/ASM ... $69

CRASMB
for OS9 & FLEX
$399

CRASMB is the highly acclaimed cross assembler package for OS9 and FLEX systems, and is the only one of its type available. It turns your computer into a development station for these CPUs:

6800 6801 6804 6805 6809 6811 6502
7000 1802 8048 8051 8080 8085 Z80
(68000 16/32 bit cross assembler ... $249)

CRASMB features include: Macros, Conditional assembly, Library file calls (12 deep), Symbol length to 30 characters. Symbol cross reference tables. Object code in 4 formats (OS9, FLEX, S1-S9, INTEL HEX), plus many other extended directives and options not found on other assemblers.

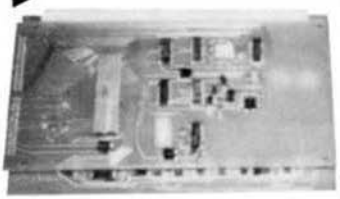
LLOYD I/O 19535 NE GLISAN, PORTLAND, OR 97230 USA
Phone: (503) 666-1097 (Software Consultation Available)

VISA, MC, COD, CHECK, APPROVED P.O.'s ACCEPTED

England: Vivaway (0582 423425), Windrush (0692 405189)
Germany: Zacher Computer (65 25 299)
Australia: Paris Radio Electronics (61 2 344 9111)

OS9 is a ™ of Microware, FLEX is a ™ of TSC

SUPER CPU ... only from LSI
68008



Here's mighty proud of our new SuperCPU card. We're giving you the 68k's to go 68000 without major changes to your system. Our new CPU gives you these enhanced features:

- Dynamic partitioning memory management unit with bound check register
- On board timer for multi user/multi tasking applications
- On board boot strap EPROM and Monitor EPROM space
- Vectored 7 level interrupt controller
- On board real time generator
- User selectable bus speeds that includes a new higher bandwidth bus mode

And many more

SuperCPU ASSEM & TESTED $499.95
SuperCPU KIT FORM $449.95
KIT INCLUDES PROCESSOR, CRYSTAL, SOCKETS AND CONNECTORS
DISK CONTROLLER SUPPORTED
DC-4 and AAA 8d

449.95

Announcing . . .

THE *SHELL* FOR FLEX 9™*

We are pleased to announce the **SHELL**, a UNIX ++ like shell that supports I/O redirection, pipes, macro substitution and programmable shell scripts! The shell will work with all your existing programs and utilities. Requires 56K of user ram. FLEX 9™* version 2.0 and above. The shell occupies the top 8K of user ram. An excellent tool for the 6809 community.

FLX/SHO9-8 8 inch version	90.00
FLX/SHO9-5 5.25 inch version	90.00
ONE YEAR MAINTENANCE	22.50

LSI STANDS BEHIND ITS PRODUCTS
1 YEAR LIMITED WARRANTY
ON ALL OUR PRODUCTS

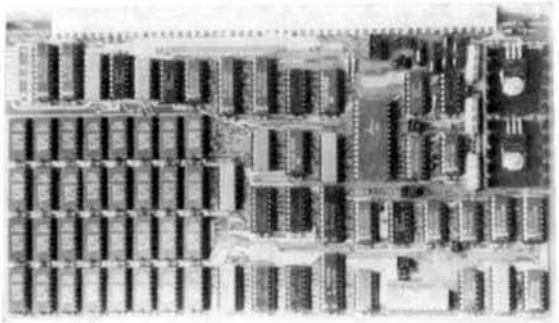
N.Y. residents add sales tax

VISA master charge

and COD accepted

*FLEX 9™ is a trademark of Technical Systems Consultants Inc.
UNIX ++ is a registered trademark of Bell Labs.
All prices and offers subject to change without notice.

LSI Enterprises Ltd.
PO Box 1227
Woodhaven, NY 11421
(212) 423-5596



256K, 512K, 1 MEG MEMORY SYSTEM

New compatible with DMA controllers. Runs at up to 2Mhz without generating MRDY or interrupts. Has an optional on board DAT for use with CPU cards without a DAT. 128K, 256K, 512K or 1M byte per card. Field upgradable. Optional configuration allows 4M byte address reach (using memory board DAT) without CPU changes or cables. 1 year limited warranty.

TURBO virtual disk software and memory diagnostics supplied with the system.

Prepaid: 256K: $749.00, 128K: $595.00, 512K: $1495.00, 1024K: $2495.00. Domestic shipping and handling $10.00. Users manual: $15.00, applicable toward system purchase. Cashiers check, COD, personal checks must clear before shipment. Fla. residents add 5% sales tax. Shipped stock to 30 days. Dealer and quantity discounts available.

COMPUTER EXCELLENCE INC.
P.O. BOX 8442
CORAL SPRINGS, FL 33065
(305) 752-8321

DATA ACQUISITION CP/M 2.2

NOW THERE IS AVAILABLE ON THE S50 BUS INDUSTRIAL QUALITY BOARDS FOR YOUR DEMANDING DATA ACQUISITION NEEDS

ADC1200

- *HIGH SPEED 12 BIT A/D BOARD*
- *16 CHANNELS Single-ended or Eight Differential*
- *25 uSEC CONVERSION TIME*
- *80k SAMPLES PER SECOND in single Channel Burst Mode*
- *INSTRUMENTATION AMPLIFIER / Resistor Selectable Gain*
- *Contained on Single 30 Pin Board*
- *CONFIGUREABLE in a Variety of Computer Controlle M es*
- *SOFTWARE EXAMPLES*
- *$795 Each $678 2 to 4*

DAC 1220

- *HIGH SPEED 12 BIT D/A BOARD*
- *TWO INDEPENDENT DIGITAL TO ANALOG CONVERTERS*
- *10 uSEC SETTLING TIME*
- *DOUBLE BUFFERED*
- *BLANKING OUTPUT PULSE*
- *FOUR QUANDRANT MULTIPLY using EXTERNAL REFERENCE Input*
- *Contained on Single 30 Pin Board*
- *$395 Each $336 2 to 4*

GPIB4800

- *IEEE 488 CONTROLLER BOARD*
- *Talk r, Liste r, Controller, Master*
- *Uses the TI 9914 Controller Chip*
- *IEEE 488 Panel Mount Connector*
- *Contained on Single 30 pin Board*
- *$295 Each 1 to 4*

Z809

- *CP/M 2.2 OPERATING SYSTEM*
- *Z80 CO-PROCESSOR*
- *ASSEMBLER DEBUGGER UTILITIES*
- *PUBLIC DOMAIN SOFTWARE*
- *$595 Each $476 2 to 4*

WRITE OR CALL TODAY FOR COMPLETE DATA

FOREIGN DEALERS
Kline Computers • D-5521-Irrel, West Germany
Tel: 652-5299
Digicamp AG • Zurich, Switzerland
Tel: 1-461-12-13
Bernstein Computer Consultants • Cape Town, South Africa • Tel: 21-6394

META LAB

(303)651-9402
6825 COUNTY LINE ROAD
LONGMONT, CO 80501

FREE DISKETTE WITH EVERY $50 PURCHASE

☎ TOLL FREE TELEX 558 414 PVT BTH

1-800-338-6800
For Ordering

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343

for information
call (615) 842-4601

CoCo OS-9™ FLEX™

SOFTWARE

Southeast Media

DIET-TRAC Forecaster

DIET-TRAC Forecaster is an XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate. Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual.

Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are t ken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. When a weight goal is given (either gain or loss), and a calorie plan is agreed upon between the computer and the individual, the number of days to reach the weight goal is projected. The starting and ending rate of weight loss is calculated, and a daily calendar with each day's weight for a 30-day period is printed.

F - $59.95
U - $89.95

Southeast Media

XDATA
A COMMUNICATION Package
for the UniFLEX Operating System

Allows UniFLEX Based Systems to Transmit and Receive files to and from other Computer Systems via Modem. Use with CP/M, Main Frames, other UniFLEX Systems, etc.

- -- Verifies Transmission integrity using checksum or CRC
- -- Automatically Re-Transmits bad blocks
- -- Transmits data in 128 byte blocks

U - $299.99

Southeast Media

JUST
Text Formatter

JUST, a Text Formatter developed by Ron Anderson, provides numerous features which make it a valuable addition to any FLEX Users Software Library. JUST is designed for formatting Text Output for Dot Matrix Printers and provides many unique features:

- -Output the "Formatted" Text to the Display for format analysis and change.
- -Output the "Formatted" Text to a Text File for use with the supplied FPRINT.CMD for producing multiple copies of the Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (this Utility is very useful at other times also, and worth the price of the program by itself).
- -"User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); provides for up to ten (10) imbedded "Printer Control Commands", such as Italics on and off, boldface on and off, etc.
- -Automatic compensation for a "Double Width" printed line.
- -Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc.
- -Use with ANY Editor.
- -Supplied with "Structured Source" (Windrush PL/9); easy to see the flow of the program.

F and CCF - $49.95

Lucidata

PASCAL UTILITIES
Requires LUCIDATA Pascal ver 3.

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

F and CCF - $25.00

INCLUDE -- allows the inclusion of other Files in a Source Text; has unlimited nesting capabilities. Also allows Binary File inclusions.

F and CCF - $25.00

PROFILER -- produces an Indented, Numbered, "Structogram" of a Pascal Source Text File. Allows viewing the overall structure of large programs, and provides clues as to the integrity of the program. Supplied as Source Code; requires compilation.

F and CCF - $25.00

Lucidata

COPYCAT
Pascal NOT required

Allows reading TSC Mini-FLEX, SSB DOS68, and Digital Research CP/M Disks while operating under FLEX 1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform Miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Includes Utilities to List Directories, Copy Files, and convert Text Files when required. Also includes a Utility for investigating Physical Compatibility problems. Programs supplied in Modular Source Code (Assembly Language) to make it easier to solve unusual problems.

F and CCF 5" - $50.00
F 8" - $65.00

Computer Systems Consultants

FLEX DISK UTILITIES

Eight (8) different FLEX Utilities that should be a part of every FLEX Users Toolbox; Assembly Language (Source Code):

Copy a File with CRC Errors, so it can possibly be salvaged; **Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on the Disk; print Disk Identification; and Sort and Replace the Disk** Directory (in sorted order).

F and CCF - $50.00

WORD PROCESSORS

Alford and Associates

SCREDITOR III

EXTREMELY Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; EXCELLENT Documentation (over 300 pages), including a full Tutorial Section to help you learn how to use the system. Features Cursor-based editing, dynamic Screen Formatting (what you see is what you get), Multi-Column display and editing, "decimal align" columns (AND add them up automatically, if wanted), define multiple keystroke macros, even and odd page number headers and footers, imbed printer control codes in text, full justification series of commands, full "help" support, store common command series on disk for future use, etc. Easy "Set-Up" (for example, you just hit the key you want to use for a specific function, such as "cursor up", and the System reads an stores that key - no digging into tech manuals for codes, etc.); use supplied "set-ups", or remap the keyboard to what you are used too. Except for proportional printing, this package will DO IT ALL!

6800 or 6809 FLEX or SSB DOS, OS-9 - $175.00

Gre t Plains Computer Co.

STYLOGRAPH

A full-screen oriented WORD PROCESSOR -- (now runs on the Data-Comp and FHL Color FLEX Systems; uses the 51 x 24 Display Screens). Full screen display and editing (i.e., what you see is what you get); supports the Daisy Wheel proportional printers.

SPECIAL CCF - $195.00
F and O - $295.00 U - $395.00

SPELL

Fast Computer Dictionary.
F, CCF, OS/9 - $125.00 U - $175.00

MAIL MERGE

Greatly extends the power and flexibility of STYLOGRAPH.
F, CCF, O - $145.00 U - $195.00

VISA MasterCard

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware

☎ TOLL FREE
1-800-338-6800
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

Great Plains Computer Co.

MAIL MERGE

Greatly extends the power and flexibility of SUPERSCRIPT. Allows Multiple Text files to be printed out as one large document. Provides for merging information into the Text File during printing (such as different names and addresses), etc.

F, CCF, O - $145.00
U - $195.00

Southeast Media

SPELL8 "Computer Dictionary"
OVER 120,000 words!

No more "Let your fingers do the walking through the Dictionary" while you are entering Text with your favorite Editor or Word Processor. SPELL8 is more than just "another Spelling Checker"; it allows you to look up a word from within your Editor or Word Processor so that you **KNOW** it is right WHEN YOU TYPE IT IN with the SPH.CMD Utility (which operates in the FLEX Utility Space). Yes, it ALSO allows you to check and update the Text after you are finished; along with allowing you to ADD WORDS to the Dictionary, "Flag" questionable words in the Text for evaluation later, "View a word in context" before changing or ignoring, etc. SPELL8 first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELL8 also allows the use of Small Disk Storage systems.

F and CCF - $129.95

Great Plains Computer Co.

SPELL

Fast Computer Dictionary -- allows directly changing the Text File, adding words to the dictionary, etc. 75,000 words in less than 400 sectors.

F, CCF, OS/9 - $125.00
U - $175.00

DATA BASE MANAGEMENT SYSTEMS

Southwestern Applied Business Systems

XDMS

Possibly one of the most powerful Database Management Systems' available, this machine language program is small enough to operate on a **single sided 5" disk**, yet provides the **speed** of M.L. and **power** limited only by the user's imagination. This DMS supports Relational, Sequential, Hierarchical, and Random Access File Structures, and has Virtual Memory capabilities for those Giant Data Bases. **XDMS Level I** provides a functional "entry level" System which provides for defining a Data Base, entering and changing the Data, and producing Reports. **XDMS Level II** adds the POWERFUL "**COMMAND**" facility which uses an English Language Command Structure to manipulating the Data to create new File Structures, Sort, Select, Calculate, etc. **XDMS Level III** adds several special "Utilities" which provide additional ease of working with the various structures, changing System Parameters, etc.

XDMS Lvl I - F & CCF - $129.95
XDMS Lvl II - F & CCF - $199.95
XDMS Lvl III - F & CCF - $269.95
XDMS System Manual only - $24.95

Great Plains Computer Co.

DBMS

An XBASIC, Menu Driven, DBMS with "Built-in" Audit Tracking, Extremely Powerful Report & Format Capabilities, etc. This **Time Proven** DBMS will become the "Work Horse" of your Software Stable.

F and CCF $295.00
U $395.00

ACCOUNTING PACKAGES

Great Plains Computer Co. and **Universal Data Research, Inc.** both have Business Packages written in TSC XBASIC for FLEX, **CoCo FLEX**, and **UniFLEX** ----

• • • • • • • • • • • •

FREE DISKETTE WITH EVERY $50 PURCHASE

☎ **TOLL FREE** TELEX 558 414 PVT BTH
1-800-338-6800
For Ordering

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

Computer Systems Consultants

BASIC UTILITY PROGRAMS

Ten BASIC Programs to:

A BASIC Resequencer with EXTRAs over "RENUM"; works with ALl Versions of FLEX BASIC **AND** the Precompiler, checks for missing label definitions, processes Disk to Disk instead of in Memory.

Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files.

A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, XBASIC, and PRECOMPILER BASIC Programs.

ALL Utilities include Source (either BASIC or Source Code). An EXCELLENT Value!

F and CCF - $25.00
UniFLEX - $50.00

Computer Systems Consultants

FULL SCREEN INVENTORY/MRP

The Full Screen Inventory System provides a means of maintaining small inventories. Using a linked, keyed random file structure based upon the item field, it keeps the file in alphabetical order for easier inquiry. With the FIND command, the user may locate and/or print all records matching on partial or complete item, description, vendor, or attributes. Items in backorder or below minimum stock levels may be located and/or printed thru the same process. Printed output may be produced in item or vendor order. A materials requirement planning (MRP) capability for manufacturing environments is included to allow the maintenance and analysis of Hierarchical assemblies of items in the inventory file. It requires TSC's Extended BASIC.

F and CCF - $100.00. U - $150.00

The Virginia Company ***Bizpack***

BIZPACK is used for storing accounting, numeric, and financial data which can then be used for planning, budgeting, forecasting, analyzing, etc. While "Electronic Spreadsheets" are extremely useful in many situations, BIZPACK excels in businesses where there are numerous expense columns, revenue sources, significant business indicators, large numbers, erratic week-to-week and month-to-month fluctuations, etc. BIZPACK helps determine statistical relationships, establish trend lines, "smooths" data via moving averages, analyze seasonal data, adjusts for inflation, lags data in Statistics or Column functions, plots data, etc. BIZPACK is oriented toward time series analysis of businesses. The Program displays information on the screen in Columns of Information with each Row conforming to a defined Period of Time (weeks, months, years, etc.), and is very easy to use (data is easy to enter, change, and modify; commands can be renamed to suit the users requirements; unlimited ability to create specialized commands using common BASIC Statements; etc.). Requires TSC's Extended BASIC.

F and CCF - $135.00
with Source - $250.00*

*** SPECIAL ***
Purchase XBASIC and BIZPACK together for $221.50
-- a Savings of $13.50 --

VISA MasterCard

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

1-800-338-6800
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
CoCo OS-9™ FLEX™
SOFTWARE
Info (615) 842-4601

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

!!! Please Specify Your Operating System & Disk Size !!!

FREE DISKETTE WITH EVERY $50 PURCHASE

*** SPECIAL ***
Purchase EDTASM and EDITPACK together for $221.50
— a Savings of $13.50 —

Computer Systems Consultants
TABULA RASA SPREADSHEET

TABULA RASA is similar to DESKTOP/PLAN and provides for the generation and maintenance of tabular computation schemes often used for analysis of business, sales, and economic scenarios. Its menu-driven user interface provides these capabilities even to those users with no programming experience. Its extensive report-generation capabilities allow the user to generate professional results with minimum effort. It requires TSC's Extended BASIC.

F and CCF - $100.00, U - $125.00

Computer Systems Center
DYNACALC

THE Electronic Spread Sheet for 6809 Computer Systems. An extremely POWERFUL Business Tool, this Program will find an unlimited number of "non-business" applications, also (for example, a Full Junior College Electronics Curriculum was set up using DYNACALC). Advanced features like "Table Lookup" make Income Tax work easy; Column or Row Sorting for numerous applications; etc. Completely "Memory Resident", Machine Language, this Program is FAST. Provides STANDARD FLEX Text File output for use with BASIC, Word Processors, Pascal, "C", etc. Also available for Data-Comp and FHL FLEX systems using the 50 x 24 Displays.

F and SPECIAL CCF - $200.00
CoCo OS9 - $99.95
O - $250.00
U - $237.00

ODDS & ENDS

Computer Systems Consultants
FULL SCREEN FORMS DISPLAY

This package supports any Serial Terminal with cursor control of Memory-Mapped Video Displays. The package substantially extends the screen Input/Output capabilities of TSC's Extended BASIC programs by providing a simple, table-driven method of describing and using full screen displays. These table entries are easy to set up and maintain, and are normally stored on disk and read as required. A simple, interactive means of generating the forms and the data field definitions is provided.

F and CCF - $50.00, U - $75.00

Computer Systems Consultants
FULL SCREEN MAILING LIST

The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Using a random fill structure based on the first character of the name field, it maintains the file in alphabetical order for easier inquiry. With the FIND command, the user may locate all records matching on partial or complete name, city, state, zip, or attributes. Printed listings and output to labels may also be produced on the same selective basis. It requires TSC's Extended BASIC.

F and CCF - $100.00, U - $110.00

COLOR COMPUTER SOFTWARE

Stearns Electronics
FORTH

Intrigued by Forth??? Here is a FORTH package tailored to the Color Computer! This package is supplied on Tape, with instructions for transferring it to disk if you wish. Written primarily in machine language, it's speed is unparalleled. A full Semigraphic-8 Editor is provided, along with "goodies" like Graphics and Sound Commands, Printer Commands, Auto-Repeat and Control Keys, etc. If you are interested in learning FORTH, a Trace Feature is provided which is invaluable. If you are a FORTH Pro, this package provides CPU carry Flag accessability, Fast Task Multiplexing, Clean Interrupt Handling, etc. (Or: you won't "out grow" the Basic capabilities of this Implementation). Combine this package with two Brodie's EXCELLENT Book "Starting FORTH", and you will be a FORTH Expert before you know it (and have a lot of fun doing it!).

Color Computer TAPE - $58.95

Custom Software Engineering, Inc.
Color Computer GRAPHIC SCREEN PRINT Programs

Dumps any "PMODE" Screen to the Printer with the BASIC USR Function. Shift the Printout Left or Right or Reverse Print (Dark for Light Screen and Vice Versa). All Programs on Tape.

GSPR	for R.S. LP-VII/VIII & DMP 100/200/400	$7.95
GSPE	for Epson w/ Graftrax and Graftrax +	$9.95
GSPG	for Gemini 10 and 15	$9.95
GSPP	for the Prowriter Printers	$9.95

Custom Software Engineering, Inc.
DATE-O-BASE CALENDAR Program

A Menu Driven EXTENDED BASIC Program which allows the entry of up to 12 Memos per Day, each of which may contain up to 28 Characters, for any day of the Month between the years 1700 and 2099. A Graphic Calendar shows which days contain Memos, and a "Key Word" Search is provided which can be output to the Screen or Printer.

TAPE DATE-O-BASE CALENDAR
(Each Tape File will hold up to 400 Memos) $16.95

DISK DATE-O-BASE CALENDAR
(4,000 Memos at 300/Month per Disk) $19.95

Custom Software Engineering, Inc.
That's Interest-ing

Interested in INTEREST (the Money Mind)? An EXTENDED BASIC Program that will help you deal with numerous problems requiring interest calculations. Present Value, Rate of Return, Current Bond Yield and Rate of Return to maturity, Loan Repayment Amortization Schedules, etc.

TAPE - $29.95

Custom Software Engineering, Inc.
DISK DATA BASE 68

An EXTENDED BASIC Data Management System w/ Mach. Lang. Routines. Allows a max of 246 Chars. and 14 Fields per Record, and another Record can be linked to the first: 8 Char. Field Names, up to 99 Chars. per Field. Powerful On-Screen editor for input and update, flexible Output capabilities including output to Disk Files for use by other Programs. Change File Definition without re-entering the Data, Split Files, etc. Allows Multiple Field Sorts, Select on any combination of fields, etc. An extremely POWERFUL TOOL! Instructions provide examples of Mailing Lists and a Financial Stock Profit and Loss Tracking System.

DISK - $54.95

Custom Software Engineering, Inc.
DISK DOUBLE ENTRY

DISK EXTENDED BASIC Accounting Program w/ Mach. Lang. Routines. A "Traditional" Accounting Package for Small Business, Clubs, Churches, Personal Use, etc. Up to four Levels of subtotals with Trial Balance, Income Statement, and Balance Sheet Reports. DDE allows up to 300 Accounts and a Trial Balance of $9,999,999.99. Transactions may be up to 14 lines long, and comments and explanations may be freely used. Accounts are traceable to the journal transaction, which may include comments. Screen reports allow review of past transactions and current Balances.

DISK - $44.95

VISA MasterCard

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

PASCAL Compilers

TSC

PASCAL Compiler

Native Code Compiler (UCSD Oriented).

F and CCF - $200.00

Lucidata

PASCAL Compiler

P-Code Compiler (ISO Standard). Designed especially for Microcomputer Systems; Run-time System checks available resources for each task, allowing operation on even minimal computer systems. Allows linkage to Assembler Code for maximum flexibility.

F and CCF 5" - $150.00
F 8" - $225.00

OmegaSoft

PASCAL Compiler

For the PROFESSIONAL; ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Use custom I/O devices in place of the Pascal INPUT and OUTPUT; Long Int. (32 Bit); Dynamic length strings; Interrupt processing, ROM-able, PIC, Re-Entrant Code, etc. POWERFUL! Includes Source for the Symbolic Debugger, Runtime, and several Utilities. Requires a "Motorola Compatible" Relocating Assembler and Linking Loader.

F and CCF - $425.00
One Year Maint. - $100.00

DECOMPILERS

Southeast Media

DUB (A UniFLEX "basic" De-Compiler)

Re-Create a Source Listing from UniFLEX Compiled basic Programs. Easy to Use: works w/ ALL Versions of UniFLEX basic; Output to Disk or Terminal. Time TESTED and PROVEN; SOLID!

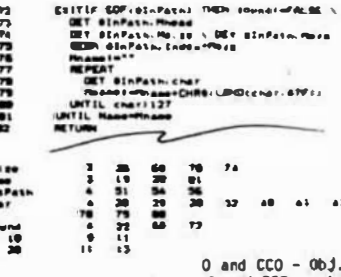
U - $219.95

UTILITIES

Southeast Media

Basic09 XRef

This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also included is a Program List Utility which outputs the listing without the overhead of building the cross reference table, which allows it to run considerably faster when only a "pretty printed" listing with line numbers is desired. Requires Basic09 or RunB for operation.



O and CCO - Obj. Only -- $39.95
O and CCO - w/ Source - $79.95

Southeast Media

OS-9 VDisk

Give your OS-9 Level I System the speed of memory access that can be several orders of magnitude over your present floppy disk drive. Use that Extended Memory capability of your SWTPC or Gimix CPU card (or any other that has the same format DAT). The size of the Virtual Disk is completely variable in whole increments of 4K up to 960K, which is all that these systems can address beyond the base page that OS-9 Level I uses. By putting all of your CMDS Directory on your Virtual Disk, you can have the fastest execution speed possible (next to eating up System Memory with all of them). You can also set up high speed inter-process communications via random virtual disk files and not eat up valuable system memory with pipe buffers. Some Assembly Required - Level I ONLY.

O. obj. only - $79.95
w/ Source - $149.95

Southeast Media

O-F

---- OS/9 to FLEX - FLEX to OS/9 ----

Finally; the barrier has been removed between OS/9 and FLEX formatted disks! Now you can READ from, and WRITE to, a Single Sided 5" or 8" FLEX diskette from OS-9 with O-F. O-F is a new and unique program, written in BASIC09 (with Source), that performs the following functions;

REFORMAT: A BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX Format so it can be used normally by FLEX.

FLEX: A BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk, all selected from a user-friendly menu. Functions provided include reading the FLEX Directory, Deleting FLEX Files, Copying both directions, etc. All selections are interactive and complete, including all necessary prompts to the operator.

FLEX users can read, write and use the special disk as any other FLEX disk, provided the FLEX directory is not allowed to continue beyond track zero (too many files).

F and CCF - $79.95

Southeast Media

COPYMULT

--- Copy LARGE Disks to several smaller disks ---

The following FLEX utilities allow the backup of ANY size disk to any SMALLER size diskettes (Winchester to 8's or 5's, 8" to 5's, etc.). By simply inserting diskettes as requested by COPYMULT, a large disk system may be downloaded to your present floppy disk system, any size. No need to fiddle with directory deletions or any of the other tedious operations that must be done using the normal copy routines.

COPYMULT.CMD understands normal "copy" syntax and always keeps up with files already copied by maintaining directories for both host and receiving disk system, eliminating hours of tedious keyboard entries and other time consuming cleanup chores.

BACKUP.CMD is a special program that downloads "random" type files, any size.

RESTORE.CMD a special program to restructure copied "random" files for copying, or recopying back to the host system.

FREELINK.CMD a "bonus" utility that "relinks" the free chain of floppy or hard disk thereby eliminating fragmentation.

Completely documented source files included.

ALL 4 Programs (8" or 5") $99.50

Southeast Media

CHESS 6809

Requires FLEX and DISPLAYS On Any Type Terminal

Features:

*Four levels of play.
*Swap side. *Point scoring system.
*Two display boards. *Change skill level.
*Solve Checkmate problems in 1-2-3-4 moves.
*Make move and swap sides. *Play white or black.

This is one of the strongest CHESS programs running on any microcomputer, estimated USCF Rating 1600+ (better than most 'club' players at higher levels).

F and CCF - $79.95

VISA MasterCard

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware

TOLL FREE 1-800-338-6800 for Ordering
SOUTH EAST MEDIA
5900 Cassandra Smith Rd. Hixson, TN 37343 Info (615) 842-4601
CoCo OS-9™ FLEX™ SOFTWARE

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

!!! Please Specify Your Operating System & Disk Size !!!

FREE DISKETTE WITH EVERY $50 PURCHASE

TOLL FREE 1-800-338-6800 For Ordering

TELEX 558 414 PVT BTH

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

ASSEMBLERS

Southeast Media

ASTRUK09

A "Structured Assembler for the 6809" which requires the TSC Macro Assembler. Allows direct use of structured statements such as IF, ELSE, DO, REPEAT, etc., and provides indented level formatting of the listing so that the structure is apparent. Re. '68' Micro Journal, Sept. '83 (program was called "STASM09"; has been renamed due to conflicts).

A User reports

"... I'm very pleased and am now writing almost exclusively in (ASTRUK09). I've selected it over --- for all future systems development... As (one) of my early evaluations, I rewrote a rather elaborate routine originally done in assembly. Out of the 1000 bytes of code generated, the (ASTRUK09) version used only 20 more bytes than the original. --- could not handle this program since it uses triple-precision fixed point arithmetic... I have a large body of code already written that is incompatible with --- constructs. No problem with (ASTRUK09) and the structure sure helps in understanding the logic!"

F, CCF - $99.95

TSC

Macro Assembler

The FLEX STANDARD Assembler. F,CCF $50.00

Relocating Assembler w/Linking Loader

Use with many of the C and Pascal Compilers. F,CCF $150.00

Great Plains Comp. Co.

RELMAC

Relocating, Recursive-Macro Assembler and Linking Loader.

F,CCF $120.00; w/Source $240.00

OmegaSoft

FRALL1

Relocating Assembler and Linking Loader

F,CCF $125.00; for One Year Maint., add $50.00

Windrush Micro Systems

MACE, by Graham Trott. F,CCF - $98.00

Computer Systems Consultants

SUPER SLEUTH

Computer Systems Consultants Super Sleuth is a "Time Tested", reliable, PROVEN Disassembler that has gained acceptance through out the SS-50 Bus Community as an extremely POWERFUL, INTERACTIVE, Software Tool. The Super Sleuth Software Package consists of 3 Programs; SLEUTH (the Disassembler), CHGNAM (used to globally Change Labels to a meaningful Name), and XREF (a Cross Reference Generator for Source Code Files). SLEUTH will Disassemble Memory Resident 6809 Code and 6800, 6801, 6802, 6803 (the "Baby CoCo"), 6805, 6808, 6809, and 6502 (Apple, Atari, Commodore, etc.) Binary Disk Files. (See Aug. '83 '68' Micro Journal "Color Users Notes" Column for a full Review.)

Color Computer	SS-50 Bus (all w/ Source)
CCD (32K Req'd)	
Obj. Only $49.00	F, $99.00
CCF, Obj. Only $50.00	U, $100.00
CCF, w/Source $99.00	O, $101.00
CCO, Obj. Only $50.00	

ALL Computer Systems Consultants Software
runs on the Color FLEX Systems
ALL in stock
call 800-338-6800
for IMMEDIATE DELIVERY

Computer Systems Center

DYNAMITE +

An "easy to use", powerful Disassembler for Disk Resident 6809 and 6800 Binary Files. Allows the development of a "Control File" of various Program "Bounderies" during successive disassemblies; can use a Label File which automatically replaces a Hex location with a Label Name; includes an XREF Utility; etc. Label Files provided for Mini-FLEX, FLEX2, FLEX9, Color Computer (for use with Color FLEX Systems), etc. OS-9 Version includes special OS-9 options.

CCF, Obj. Only $100.00
CCO, " " $59.95
F, " " $100.00
O, " " $150.00
U, " " $300.00

COMPILERS & DECOMPILERS

6809 "Structured" Assembly Lang. Compilers

Windrush Micro Systems

PL/9

By Graham Trott. A combination Editor/Compiler/Debugger, all in ONE PACKAGE; provides a totally INTERACTIVE Program Development Cycle. The Single-Pass Compiler supports large Symbol Names; Variable Types; Pointers; Control Structures (similar to 'C' or 'Pascal'); Stack, A- ,B-, and D-Register manipulation; etc. The Source-Oriented Trace/Debugger provides Single Stepping, Break-pointing, etc. An excellent Software Development Tool which provides for the maximum utilization of the power of the 6809.

F, CCF - $198.00

Whimsical Developments

WHIMSICAL

Need the Ease of Design and Maintainability of "Structured Programming" AND the Speed and Control of Assembly Language? Then WHIMSICAL was designed for you! This Single Pass, Recursive Descent Compiler provides the tool for developing simple Utilities to MAJOR Systems in Assembly Language. Supports 3 "Lex" Levels which allow one level of Procedure nesting, or more within "Modules". It is easy to develop programs written for other machines since you are working at the Assembly Language level. Features unified, user-defined I/O; produces ROMable, relocatable, recursive, re-entrant Code; Structured style and statements with Procedures and Modules; supports Byte and Double-Byte primitives with 3 types of Integers (up to 32 bit), Char and Boolean, and unlimited sized Arrays (vectors only); Interrupt handling; unlimited length Variable Names; Variable Initialization (defaults to $00); include "Source File" directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. To quote Ron Anderson in his review of WHIMSICAL in the Sept. '83 issue of '68' Micro Journal that, except for the lack of floats, "..., I have to give this one VERY high rating, ...". It is a FAST Compiler which produces FAST Code (his "Primes" Benchmark ran at 9 secs. on a 2 MHz System).

F and CCF - $195.00

'C' Compilers

Windrush Micro Systems

C Compiler

By James McCosh. Full featured C Compiler for the FLEX Operating System (lacking ONLY "bit-fields"), including an Assembler. Requires the TSC Relocating Assembler IF the user wishes to implement his own Libraries.

F and CCF - $295.00

Introl

C Compiler

A full-featured C, streamlined for the 6809. Generates very efficient object code. Output "Benchmarks" close to 10 MHz 68000 in 8 Bit Operations; 1.5 times faster than a 4 MHz Z80 when using a 2MHz 6809 System (Re. p 43, '68' Micro Journal, May '83). Floats, etc.

F, CCF, and O - $375.00
U - $425.00
One Year Maint. - $100.00

VISA MasterCard

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

TOLL FREE 1-800-338-6800 For Ordering
SOUTH EAST MEDIA
5900 Cassandra Smith Rd. Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends —

F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

!!! Please Specify Your Operating System & Disk Size !!!

Computer Systems Center

DYNASHARE

— Multi-User, Multi-Tasking with FLEX —

Southeast Media is now shipping DYNASHARE FROM STOCK - the multi-user, multi-tasking capability of DYNASHARE allows FLEX users the advantages of more sophisticated and time saving computer usage without having to buy or learn a new language or Operating System syntax. DYNASHARE, as its name implies, allows true "time-sharing" operation under the popular FLEX operating system, and also allows each user to run two simultaneous jobs (multi-tasking); even on single-user systems. For example, while in EDIT, you can list another file or examine a directory. Or, you might look up an item in a Data Base while a Sort is in progress! DYNASHARE also provides some fringe benefits that will be greatly appreciated by FLEX users, including type-ahead, command line editing, and instant response to "escape".

DYNASHARE is the painless method! Use your existing Flex computer by simply adding 64K of RAM for each user and/or task. Fact is, you still use FLEX just like you always have! DYNASHARE is not intended as competition to UniFLEX. It does not improve on the speed of FLEX, and does not offer password protection or other niceties of a full-blown multi-user system. What DYNASHARE does do is give FLEX users a low-cost way to use existing software in a multi-user, multi-tasking environment, so your existing FLEX versions of BASIC, XBASIC, editors, assemblers, disassemblers, sort/merge packages, word processors, compilers, DYNACALC spread-sheet package, and so on are still good.

NOTE -- The initial release of DYNASHARE is for SWTPC 8/09 Computers, but versions will also be available for other popular extended-memory (up to 1024K) systems, such as HELIX and GIMIX. A minimum of 128K of RAM will be required with ALL versions. DYNASHARE requires 64K of RAM for each active task; thus a 256k system could allow foreground-background operation on two terminals, or foreground-only operation on four terminals.

AVAILABLE NOW from Southeast Media - $2[illegible].00

FREE DISKETTE WITH EVERY $50 PURCHASE

TOLL FREE 1-800-338-6800 For Ordering

TELEX 558 414 PVT BTH

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343

for information
call (615) 842-4601

CoCo OS-9™ FLEX™

SOFTWARE

In the past there has been too much software offered that was not quite ready. We will strive to eliminate that element. But, right up front, we tell you only that we will do our very best; nothing more. Also, we will strive to keep cost to a bare minimum, while securing for the author a fair return in

CRUNCH COBOL
Cobol Language Compiler
(Reviewed in Nov. 1984 68' Micro Journal)
FLEX only Regular $199.00

Special Introduction $99.95

K-BASIC
Basic Language Compiler
(Reviewed in Oct. 1984 68' Micro Journal)
Includes OSM Assembler
Works with CRASMB
FLEX & OS-9 $199.00

CRASMB
6809 Cross Assembler for the following CPU types

6800-2-8	6801-3	6804	6805
6809	6811	6502	1802
8048	8080-5	Z-8	Z-80

FLEX & OS-9 $399.00

CRASMB 16.32
6809 Cross Assemble for the 68000 CPU
FLEX & OS-9 $249.00

OSM
6809 Extended Macro Assembler
(Included with K-BASIC)
FLEX & OS-9 $99.00

will qualify under this program, please contact one of the people below. Remember, if your software has any problems or "funnies" — **GET IT STRAIGHT BEFORE YOU CONTACT US!!** Also get your source code in proper shape and well commented; there is too much 99¢ code already drifting around.

If your software is READY contact:
Bob Bay, Don Williams, or Tom Williams

Southeast Media is a division of Computer Publishing, Inc. (CPI), a family of 100% 68XX support facilities.

VISA

MasterCard

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware

TOLL FREE 1-800-338-6800 For Ordering
SOUTH EAST MEDIA
5900 Cassandra Smith Rd. Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™ SOFTWARE

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

TEN MOST-ASKED QUESTIONS about DYNACALC™

THE ELECTRONIC SPREAD-SHEET FOR 6809 COMPUTERS

1. **What is an electronic spread-sheet, anyway?**
   Business people use spread-sheets to organize columns and rows of figures. DYNACALC simulates the operation of a spread-sheet without the mess of paper and pencil. Of course, corrections and changes are a snap. Changing any entered value causes the whole spread-sheet to be re-calculated based on the new constants. This means that you can play, 'what if?' to your heart's content.

2. **Is DYNACALC just for accountants, then?**
   Not at all. DYNACALC can be used for just about any type of job. Not only numbers, but alphanumeric messages can be handled. Engineers and other technical users will love DYNACALC's sixteen-digit math and built-in scientific functions. You can build worksheets as large as 256 columns or 256 rows. There's even a built-in sort command, so you can use DYNACALC to manage small data bases — up to 256 records.

3. **What will DYNACALC do for ME?**
   That's a good question. Basically the answer is that DYNACALC will let your computer do just about anything you can imagine. Ask your friends who have VisiCalc™, or a similar program, just how useful an electronic spread-sheet program can be for all types of household, business, engineering, and scientific applications. Typical uses include financial planning and budgeting, sales records, bills of material, depreciation schedules, student grade records, job costing, income tax preparation, checkbook balancing, parts inventories, and payroll. But there is no limit to what YOU can do with DYNACALC.

4. **Do I have to learn computer programming?**
   NO! DYNACALC is designed to be used by non-programmers, but even a Ph.D. in Computer Science can understand it. Even experienced programmers can get jobs done many times faster with DYNACALC, compared to conventional programming. Built-in HELP messages are provided for quick reference to operating instructions.

5. **Do I have to modify my system to use DYNACALC?**
   Nope. DYNACALC uses any standard 6809 configuration, so you don't have to spend money on another CPU board or waste time learning another operating system.

6. **Will DYNACALC read my existing data files?**
   You bet! DYNACALC has a beautifully simple method of reading and writing data files, so you can communicate both ways with other programs on your system, such as the Text Editor, Text Processor, Sort/Merge, STYLOGRAPH™ word processor, RMS™ data base system, or other programs written in BASIC, C, PASCAL, FORTRAN, and so on.

7. **How fast is DYNACALC?**
   Very. Except for a few seldom-used commands, DYNACALC is memory-resident, so there is little disk I/O to slow things down. The whole data array (worksheet) is in memory, so access to any point is instantaneous. DYNACALC is 100% 6809 machine code for blistering speed.

8. **Is there a version of DYNACALC for MY system?**
   Probably. You need a 6809 computer (32k minimum) with FLEX™, UniFLEX™, or OS-9™ operating system. You also need a decent crt terminal, one with at least 80 characters per line, and direct cursor addressing. If your terminal isn't smart enough for DYNACALC, you probably need a new one anyway. The UniFLEX and OS-9 versions of DYNACALC allow you to mix different brands of terminal on the same system. There's also a special version of DYNACALC for Color Computers equipped with FLEX (Frank Hogg or Data-Comp versions).

9. **How much does DYNACALC cost?**
   The FLEX versions are just $200 per copy; UniFLEX version $395; OS-9 version (works with LEVEL ONE or LEVEL TWO) $250. Orders outside North America add $7 per copy for postage. We encourage dealers to handle DYNACALC, since it's a product that sells instantly upon demonstration. Call or write on your company letterhead for more information.

10. **Where do I order DYNACALC?**
    See your local DYNACALC dealer, or order directly from CSC at the address below. We accept telephone orders from 10 am to 6 pm, Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Please specify diskette size for FLEX or OS-9 versions. Software serial number is required for the UniFLEX version.

Order your DYNACALC today!

Computer Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017
(314) 576-5020

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics, 161 Bunnerong Road (PO Box 380) Kingsford, 2032 NSW Australia. Telephone: 02-344-9111.

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-25-30-00.

UniFLEX software prices include maintenance for the first year.

DYNACALC is a trademark of Computer Systems Center

VisiCalc is a trademark of VisiCorp.
STYLOGRAPH is a trademark of Great Plains Computer Co.
RMS is a trademark of Washington Computer Services.
FLEX and UniFLEX are trademarks of TSC.
OS-9 is a trademark of Microware and Motorola.

WINDRUSH MICRO SYSTEMS

UPROM II

PROGRAMS and VERIFIES: 2758, 2508, 2716, 2516, 2732/2732a, MCM68764/6, 2764/2764A, 2564, 27128/27128A, and 27256. I=Intel, TI=Texas, M=Motorola.

NO PERSONALITY MODULES REQUIRED!

TRI-VOLT EPROMS ARE NOT SUPPORTED

INTEL's intelligent programming (te) implemented for Intel 2764, 27128 and 27256 devices. Intelligent programming reduces the average programming time of a 2764 from 7 minutes to 1 minute 15 seconds (under FLEX) with greatly improved reliability.

Fully enclosed pod with 5' of flat ribbon cable for connection to the host computer MC6821 PIA interface board.

MC6809 software for FLEX and OS9 (Level 1 or 2, Version 1.2).

BINARY DISK FILE offset loader supplied with FLEX, MDOS and OS9.

Menu driven software provides the following facilities:

a. FILL ..... a selected area of the buffer with a HEX char.
b. MOVE ..... blocks of data.
c. DUMP ..... the buffer in HEX and ASCII.
d. FIND ..... a string of bytes in the buffer.
e. EXAMINE/CHANGE . the contents of the buffer.
f. CRC ..... checksum a selected area of the buffer.
g. COPY ..... a selected area of an EPROM into the buffer.
h. VERIFY ..... a selected area of an EPROM against the buffer.
i. PROGRAM ..... a selected area of an EPROM with data in the buffer.
j. SELECT ..... a new EPROM type (return to types menu).
k. ENTER ..... the system monitor.
l. RETURN ..... to the operating system.
l. EXECUTE ..... any DOS utility (only in FLEX and OS9 versions).

FLEX AND OS9 VERSIONS AVAILABLE FROM GIMIX. SSB/MDOS CONTACT US DIRECT.

PL/9

- Friendly interactive environment where you have INSTANT access to the Editor, the Compiler, and the Trace-Debugger, which, amongst other things, can single step the program a SOURCE line at a time. You also have direct access to any FLEX utility and your system monitor.
- 375+ page manual organized as a tutorial with plenty of examples.
- Fast SINGLE PASS compiler produces 8K of COMPACT and FAST 6809 machine code output per minute with no run-time overheads or license fees.
- Fully compatible with TSC text editor format disk files.
- Signed and unsigned BYTES and INTEGERS, 32-bit floating point REALs.
- Vectors (single dimension arrays) and pointers are supported.
- Mathematical expressions: (+), (-), (*), (/), modulus (%), negation (-)
- Expression evaluators: (=), (<>), (<), (>), (>=), (<=)
- Bit operators: (AND), (OR), (EOR/XOR), (NOT), (SHIFT), (SWAP)
- Logical operators: (.AND), (.OR), (.EOR/XOR)
- Control statements: IF..THEN..ELSE, IF..CASE1..CASE2..ELSE, BEGIN..END, WHILE.., REPEAT..UNTIL, REPEAT..FOREVER, CALL, JUMP, RETURN, BREAK, GOTO.
- Direct access to (ACCA), (ACCB), (ACCD), (XREG), (CCR) and (SPACK).
- FULLY supports the MC6809 RESET, NMI, FIRQ, IRQ, SWI, SWI2, and SWI3 vectors. Writing a self-starting (from power-up) program that uses ANY, or ALL, of the MC6809 interrupts is an absolute snap!
- Machine code may be embedded in the program via the 'GEN' statement. This enables you to code critical routines in assembly language and embed them in the PL/9 program (see 'MACE' for details).
- Procedures may be passed and may return variables. This makes them functions which behave as though they were an integral part of PL/9.
- Several fully documented library procedure modules are supplied: IOSUBS, BIYIO, MATH32, HEXIO, FLEXIO, SCIPACK, STRSUBS, BASTRING, and REALCOM.

'... THIS IS THE MOST EFFICIENT COMPILER I HAVE FOUND TO DATE.'

Quoted from Ron Andersons FLEX User Notes column in '68. Need we say more?

MACE/XMACE/ASM05

All of these products feature a highly productive environment where the editor and the assembler reside in memory together. Gone are the days of tedious disk load and save operations while you are debugging your code.

- Friendly interactive environment where you have instant access to the Editor and the Assembler, FLEX utilities and your system monitor.
- MACE can also produce ASMPROCs (GEN statements) for PL/9 with the assembly language source passed to the output as comments.
- XMACE is a cross assembler for the 6800/1/2/3/8 and supports the extended mnemonics of the 6303.
- ASM05 is a cross assembler for the 6805.

D-BUG

LOOKING for a single step tracer and mini in-line disassembler that is easy to use?? Look no further, you have found it. This package is ideal for those small assembly language program debugging sessions. D-BUG occupies less than 6K (including its stack and variables) and may be loaded anywhere in memory. All you do is LOAD IT, AIM IT and GO! (80 col VDUs only).

McCOSH 'C'

This is as complete a 'C' compiler as you will find on any operating system for the 6809. It is completely compatible with UNIX VII and only lacks 'bit-fields' (which are of little practical use in an 8-bit world!).

- Produces very efficent assembly language source output with the 'C' source optionally interleaved as comments.
- Built-in optimizer will shorten object code by about 11%.
- Supports interleaved assembly language programs.
- INCLUDES its own assembler, the TSC relocating assembler is only required if you want to generate your own libraries.
- The pre-processor, compiler, optimizer, assembler and loader all run independently or under the 'CC' executive. 'CC' makes compiling a program to executable object as simple as typing in 'CC,HELLO.C <RETURN>'.

IEEE-488

- SUPPORTS ALL PRINCIPAL MODES OF THE IEEE-488 (1975/8) BUS SPECIFICATION:
  - Talker
  - Listener
  - System Controller
  - Serial Poll
  - Parallel Poll
  - Group Trigger
  - Single or Dual Primary Address
  - Secondary Address
  - Talk only ... Listen only
- Fully documented with a complete reprint of the KILOBAUD article on the IEEE bus and the Motorola publication 'Getting aboard the IEEE Bus'.
- Low level assembly language drivers suitable for 6800, 6801, 6802, 6809, 6808 and 6839 are supplied in the form of listings. A complete back to back test program is also supplied in the form of a listing. These drivers have been extensively tested and are GUARANTEED to work.
- Single S-30 board (4, 8 or 16 addresses per port), fully socketed, gold plated bus connectors and IEEE interface cable assembly.

PRICES

D-BUG	(6809 FLEX only)	$ 75.00
MACE	(6809 FLEX only)	$ 75.00
XMACE	(6809 FLEX only)	$ 98.00
ASM05	(6809 FLEX only)	$ 98.00
PL/9	(6809 FLEX only)	$198.00
'C'	(6809 FLEX only)	$295.00
IEEE-488	with IEEE-488 cable assembly	$298.00
UPROM-II/U	with one version of software (no cable or interface)	$395.00
UPROM-II/C	as above but complete with cable and S-30 interface	$545.00
CABLE	5' twist-n-flat 50 way cable with IDC connectors	$ 35.00
S-30 INT	S-30 interface for UPROM-II	$130.00
EXOR INT	Motorola EXORbus (EXORciser) interface for UPROM-II	$195.00
UPROM SFT	Software drivers for 2nd operating system. Specify FLEX or OS9 AND disk size!	$ 35.00
UPROM SRC	Assembly language source (contact us direct)	

ALL PRICES INCLUDE AIR MAIL POSTAGE

Terms: CWO. Payment by Int'l Money Order, VISA or MASTER-CARD also accepted.

WORSTEAD LABORATORIES, NORTH WALSHAM, NORFOLK, ENGLAND. NR28 9SA.

TEL: 44 (692) 404086
TLX: 975548 WMICRO G

WE STOCK THE FOLLOWING COMPANIES PRODUCTS: GIMIX, SSB, FHL, MICROWARE, TSC, LUCIDATA, LLOYD I/O, & ALFORD & ASSOCIATES.

FLEX (tm) is a trademark of Technical Systems Consultants, OS-9 (tm) is a trademark of Microware Systems Corporation, MDOS (tm) and EXORciser (tm) are trademarks of Motorola Incorporated.

FEATURES THE POWERFUL, THIRD GENERATION, MOTOROLA 6809 PROCESSOR!

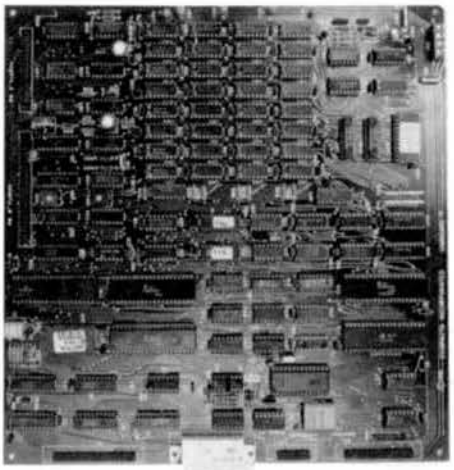
THE 6809 "UNIBOARD"™

SINGLE BOARD COMPUTER KIT

PERFECT FOR COLLEGES, OEM'S, INDUSTRIAL AND SCIENTIFIC USES!

64K RAM! DOUBLE DENSITY FLOPPY DISK CONTROLLER!

New! *Lower Price!*



BLANK PC BOARD
$99.95
WITH PAL'S, AND TWO EPROMS.
FOR 5-1/4 OR 8 INCH SOURCE DISKETTE ADD $10.

$289.00
COMPLETE KIT! FULLY SOCKETED.

PRICE CUT!!

THE COMPACTA UNIBOARD™: Through special arrangement with COMPACTA INC., we are proud to have been selected the exclusive U.S. Mfg. of their new 6809 UNIBOARD™ COMPUTER KIT. Many software professionals feel that the 6809 features probably the most powerful instruction set available today on ANY 8 bit micro. Now, at last, all of that immense computing power is available at a truly unbelievably low price.

ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A FREE COPY IS AVAILABLE UPON REQUEST.

FEATURES:

- ★ 64K RAM using 4116 RAMS.
- ★ 6809E Motorola CPU.
- ★ Double Density Floppy Disk Controller for either 5-1/4 or 8 inch drives. Uses WD1793.
- ★ On board 80 x 24 video for a low cost console. Uses 2716 Char. Gen. Programmable Formats. Uses 6845 CRT Controller.
- ★ ASCII keyboard parallel input interface. (6522)
- ★ Serial I/O (6551) for RS232C or 20 MA loop.
- ★ Centronics compatible parallel printer interface. (6522)
- ★ Buss expansion interface with DMA channel. (6844)
- ★ Dual timer for real time clock application.
- ★ Powerful on board system monitor (2732). Features commands such as Go To, Alter, Fill, Move, Display, or Test Memory. Also Read and Write Sectors. Boot Normal, Unknown, and General Flex™.

YOUR CHOICE OF POPULAR DISK OPERATING SYSTEMS:

FLEX™ from TSC	$149
OS9™ from Microware	$199
Specify 5-1/4 or 8 inch	

PC BOARD IS DOUBLE SIDED, PLATED THRU SOLDER MASKED, 11 x 11-1/2 IN.

Digital Research Computers
(OF TEXAS)
P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 225-2309

TERMS: Shipments will be made approximately 3 to 6 weeks after we receive your order. VISA, MC, cash accepted. Add $4.00 shipping. USA AND CANADA ONLY

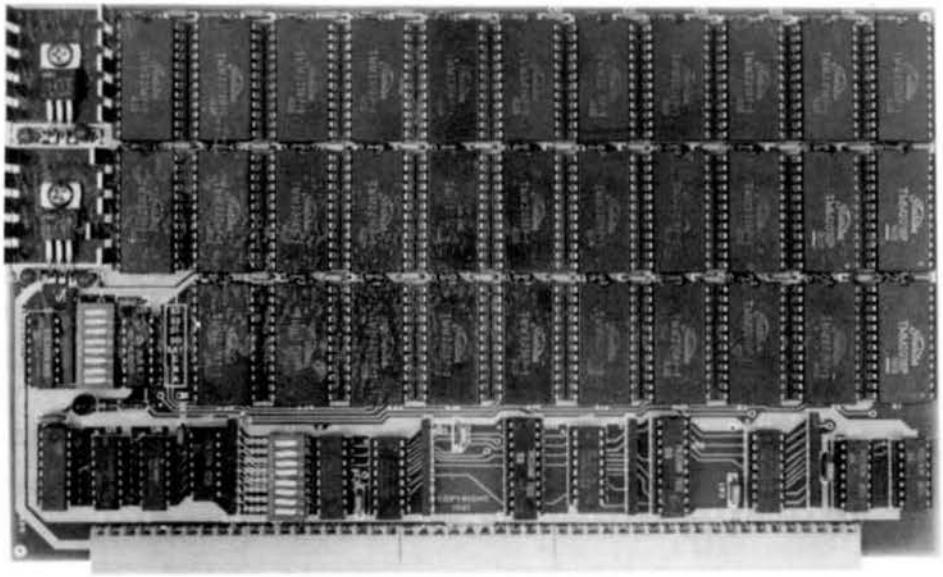
64K SS-50 STATIC RAM

PRICE CUT!!

$159.00 (48K KIT)

NEW!

LOW POWER!



RAM OR EPROM!

BLANK PC BOARD WITH DOCUMENTATION $45

SUPPORT ICs + CAPS -	$18.00
FULL SOCKET SET -	$15.00

ASSEMBLED AND TESTED **ADD $50**

FEATURES:

- ★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- ★ Fully supports Extended Addressing.
- ★ 64K draws only approximately 500 MA.
- ★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- ★ Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- ★ 2716 EPROMs may be installed anywhere on Board.
- ★ Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- ★ One Board supports both RAM and EPROM.
- ★ RAM supports 2MHZ operation at no extra charge!
- ★ Board may be partially populated in 16K increments.

56K	$189
64K	$219

CLOSE OUT SPECIAL

WE HAVE DROPPED OUR 32K SS-50 STATIC RAM BOARD WHICH USED 2114 LOW POWER RAMS. WE WILL SELL THE REMAINING STOCK OF BLANK PCB'S WITH DATA FOR $17.50 EA. THESE FORMERLY SOLD FOR $50.

16K STATIC RAMS?

The new 2K x 8, 24 PIN, static RAMs are the next generation of high density, high speed, low power, RAMs. Pioneered by such companies as HITACHI and TOSHIBA, and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts, feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality, and you get BLINDING speed and LOW power thrown in for virtually nothing.

Digital Research Computers
(OF TEXAS)
P.O. BOX 461565 • GARLAND, TEXAS 75048 • (214) 225-2309

TERMS: Add $2.00 postage. We pay balance. Order under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50, add 85¢ for insurance.

!!! FREE !!!

Published Monthly by Computer Publishing Inc., Hixson, TN.

$1.95

Bulk Rate
U.S. Postage
PAID
Chattanooga, TN
Permit No. 357

Color Micro Journal

The Color Computer Monthly Magazine

$1.95 per issue Vol. 1, Issue 2 October, 1983

THIS 'N THAT

The BIG NEWS this month is that OS-9 has finally arrived for the Color Computer. The ASTOUNDING part of the Radio Shack OS-9 Package, besides the price, is the [illegible]. You 'Old Time Radio Shack Followers' will not believe what you see. Jon Shirley has been telling us that the main reason for the "lack" of documentation with a lot of their products was the restrictions placed on releasing that information by Microsoft; I

OS-9 on the COLOR COMPUTER

One of the "Operating Systems of the Future" is now available for the "little old Color Computer": OS-9. Freely translated, OS-9 means "Operating System for the 6809" (OS-9 is now being written for the 68000, also). Since it is fairly obvious that UNIX and "UNIX-Type" Operating Systems will be running on just about every computer to come out in the next few years, a whole new language is beginning to appear on the horizon.

Color Computer OS-9, the Package

We had been running a preliminary release of OS-9 on the Color Computer for a few weeks, and received the "Official Radio Shack" version for Review a couple of days ago. To put it mildly, this package is IMPRESSIVE! For $69.95 (Radio Shack Catalog Number 26-3038), you receive a 9 1/2" x 7 5/8" x 2" package containing 4

Introducing NuBASE... the *uncomplicated* data base system.

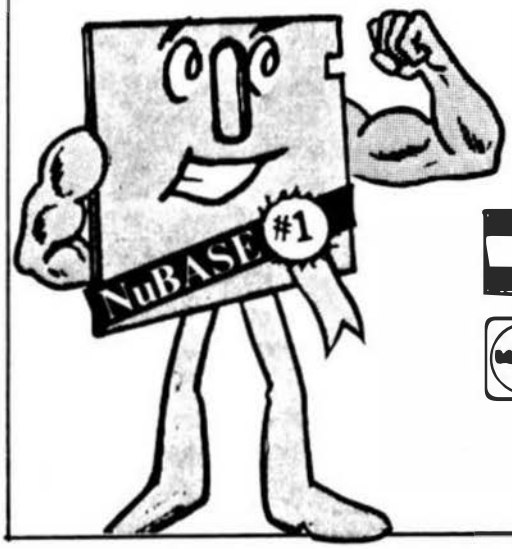
It lets you throw away all the books!

NuBASE is a DB manager so versatile that you can use it to do what you want with your data. It's not complicated or overbearing; in fact it's so easy to use, you'll be up and running in minutes.

Simple user-specified masks insure data accuracy. Data integrity is assured through the use of highly crash-resistant software. *See what you're doing* through the interactive generation of files, screens and reports.

NuBASE is as affordable as it is complete. There's nothing else to buy... $150 brings you the comprehensive package, including a ready-to-use mailing list application to get your **NuBASE** working for you on day one.

The computing power of NuBASE is limited only by the capacity of your hardware.



Currently available for OS-9 Level II

For more information or to place an order, contact:

Dept. 68 15
The JBM Group, Inc.
Continental Business Center
Front & Ford Streets
Bridgeport, PA USA 19405
TWX: 510-660-3999

215-275-1777

PA res. add 6% sales tax.
US orders, add $5.00 postage and handling.

OS9 is a registered trademark of Microware Corp.

'68' MICRO JOURNAL ADVERTISERS INDEX

This Index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

U.S. Postal Service Statement of Ownership, Management and Circulation (Required by 39 U.S.C. 3685): 1. Title of Publication: 68 Micro Journal 2. Publication no: 468510: 2. Date of Filing: 09-28-85: 3. Frequency of Issue: Monthly: 4. No. of Issues Published Annually: 12. 5 Annual Subscription Price: $24.50 4 and 5: Complete Mailing Address of Known Office, Headquarters or General Business Offices of the Publisher: 5900 Cassandra Smith Rd., Hixson, TN. 37343 6. Full Name and Complete Mailing Address of Publisher: Donald M. Williams Sr., 5900 Cassandra Smith Rd., Hixson, TN 37343. Editor: Donald M. Williams Sr., 5900 Cassandra Smith Rd., Hixson, TN 37343., Managing Editor: Larry E. Williams, 5900 Cassandra Smith Rd., Hixson, TN 37343 7.Owner: Computer Publishing Inc., 5900 Cassandra Smith Rd., Hixson, TN 37343, whose Stockholders are: Donald M. Sr., Frances J., Larry E., Mary E., Thomas E. Williams. 8. Known Bondholders, Mortgagees, and other Security Holders Owning or Holding 1 Percent or more of Total Amount of Bonds, Mortgages or other Securities: None 9.For Completion By Nonprofit Organizations Authorized To Mail At Special Rates: N/A 10. Extend and Nature of Circulation: A. Total No. Copies (Net Press Run)Average No. Copies Each Issue During Preceding 12 Months: 9142., Actual No. Copies Of Single Issue Published Nearest To Filing Date: 8500. B. Paid Circulation: 1. Sales Through Dealers And Carriers, Street Vendors And Counter Sales: Average No. Copies Each Issue During Preceding 12 Months: 4065. Actual No. Copies Of Single Issue Published Nearest To Filing Date: 4065. 2. Mail Subscriptions: Average No. Copies Each Issue During Preceding 12 Months: 4496. Actual No. Copies Of Single Issue Published Nearest To Filing Date: 4190. C. Total Paid Circulation: Average No. Copies Each Issue During Preceding 12 Months: 8559. Actual No. Copies Of Single Issue Published Nearest To Filing Date: 8293. D. Free Distribution By Mail, Carrier Or Other Means Including Samples, Complimentary, And Other Free Copies: Average No. Copies Each Issue During Preceding 12 Months: 199. Actual No. Copies Of Single Issue Published Nearest To Filing Date: 47. E. Total Distribution (Sum Of C And D): Average No. Copies Each Issue During Preceding 12 Months: 8758, Actual No. Copies Of Single Issue Published Nearest To Filing Date: 8500. F. Copies Not Distributed: 1. Office Use, Left Over, Unaccounted,Spoiled After Printing: Average No. Copies Each Issue During Preceding 12 Months: 275, Actual No. Copies Of Single Issue Published Nearest To Filing Date: 200. 2. Return From News Agents: Average No. Copies Each Issue During Preceding 12 Months: 109, Actual No. Copies Of Single Issue Published Nearest To Filing Date: 0. G. Total(Sum of E, F1 and 2-Should Equal Net Press Run Shown in A: Average No. Copies Each Issue During Preceding 12 Months: 9142, Actual No. Copies Of Single Issue Published Nearest To Filing Date: 8500. 11. I Certify That The Statements Made By Me Above Are Correct And Complete. (Signed): MARY E. ROBERTSON, Office Manager,68 Micro Journal. Computer Publishing, Inc. 12. For completion by publishers mailing at the regular rates: Permission requested.

GIMIX HAS THE 6809 SYSTEM TO SUIT YOUR NEEDS

HARDWARE

All systems feature the **GIMIX CLASSY CHASSIS**; with a ferro-resonant constant voltage power supply, gold plated bus connectors, and plenty of capacity for future expansion.

Static **RAM** and double-density **DMA** floppy disk controllers are used exclusively in all systems.

All systems are guaranteed for 2 MHz operation and include complete hardware and software documentation, necessary cables, filler plates, etc.

Systems are assembled using burned-in and tested boards, and all disk drives are tested and aligned by **GIMIX**.

You can add additional components to any system when ordering, or expand it in the future by adding **RAM, I/O,** etc.

GIMIX lets you choose from a wide variety of options to customize your system to your needs.

SOFTWARE

All **OS-9/FLEX** systems allow you to software select either operating system.

Also included is the **GMXBUG** monitor and, in systems with 128K or more of RAM, **GMX-VDISK** for **FLEX**.

All **GIMIX OS-9** systems include **Microware's Editor, Assembler, Debugger, Basic09,** and **Runb**; and the GMX versions of **RMS** and **DO** for **OS-9**.

All **GIMIX** versions of **OS-9** can read and write RS color computer format OS-9 disks, as well as the Microware/GIMIX standard format.

New and exclusive with **OS-9 GMX III** systems is the **GMX OS-9 Support ROM**, a monitor for **OS-9** that includes memory diagnostics and allows the system to boot directly from either hard disk or floppy.

A wide variety of languages and other software is available for use with either **OS-9** or **FLEX**.

OS-9 GMX III/FLEX SYSTEMS (#79)

The **#79** super system now includes (in addition to the above): the GMX **6809 CPU III**, a **256K CMOS Static RAM Board (#72)**, and a **3-port Intelligent Serial I/O Processor (#11)**.

The GMX 6809 CPU III can perform high-speed DMA transfers from memory to memory and uses memory attributes and illegal instruction trapping to protect the system and users from program crashes. If a user program crashes, only that user is affected; other users are unaware of the problem.

The 3-Port Intelligent Serial I/O Board (#11) significantly reduces system overhead by handling routine I/O functions; freeing the host CPU for running user programs. This improves overall system performance and allows user terminals to be run at up to 19.2K baud.

with dual 40 track DSDD drives	$5998.79
with dual 80 track DSDD drives	$6198.79
with **#88** dual 8" DSDD drive system	$7698.79
with **#90** 19MB Winchester subsystem and one 80 track	$8898.79
with a 47MB Winchester subsystem and one 80 track	$10,898.79
with a 47MB plus a 6MB removable pack Winchester subsystem and one 80 track drive	$12,398.79

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account #73-32033.

OS-9 GMX I / FLEX SYSTEMS #49

The **#49** systems include 64KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	$3998.49
with dual 80 track DSDD drives	$4198.49
with **#88** dual 8" DSDD drive system	$5698.49
with **#90** 19MB Winchester subsystem and one 80 track	$6898.49

OS-9 GMX II / FLEX SYSTEMS #39

The **#39** systems include 128KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	$4498.39
with dual 80 track DSDD drives	$4698.39
with **#88** dual 8" DSDD drive system	$6198.39
with **#90** 19MB Winchester subsystem and one 80 track	$7398.39

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

EXPORT MODELS: ADD $30 FOR 50Hz. POWER SUPPLIES.

GIMIX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

ALL PRICES ARE F.O.B. CHICAGO

Contact GIMIX for price and availability of UniFLEX and UniFLEX GMXIII Systems.

NOTE on all drive systems: Dual 40 track drives have about 700KB of formatted capacity; dual 80's about 1,400KB; dual 8" about 2,000KB. The formatted capacity of hard disks is about 80% of the total capacity.

Want to expand your system to a megabyte of Static RAM and 15 users?

Simply add additional memory and I/O boards. Your GIMIX system can grow with your needs. Contact us for a complete list of available boards and options.

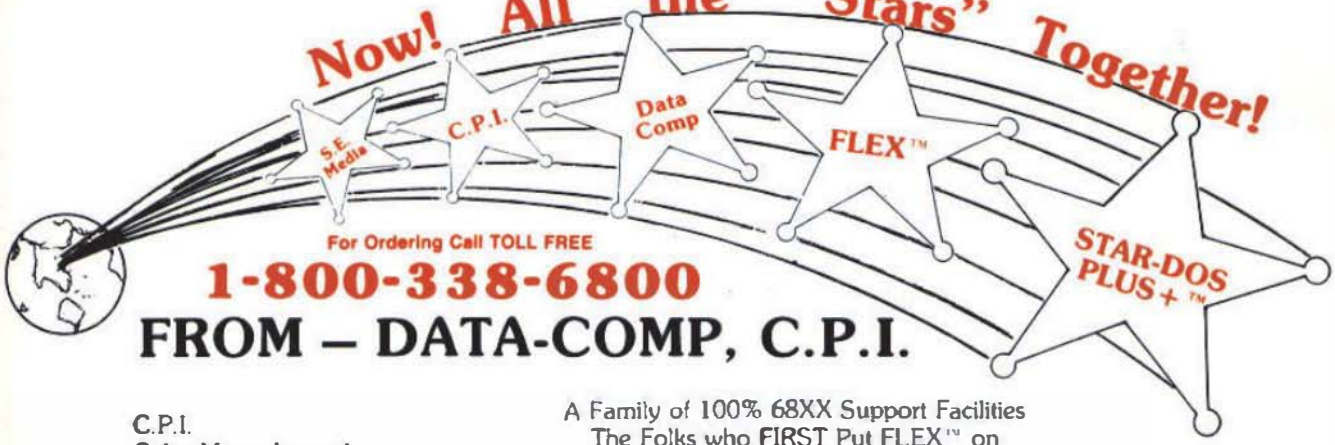
#72 256KB CMOS STATIC RAM board with battery back up	$1898.72
#64 64KB CMOS STATIC RAM board with battery back up	$528.64
#67 64KB STATIC RAM board	$476.87
#11 3 port intelligent serial I/O board	$498.11
#43 2 port serial I/O board	$128.43
#42 2 port parallel I/O board	$88.42
#95 cable sets (1 needed per port), specify board	$24.95

TRADE UP YOUR CoCo!

GIMIX will allow you up to $1100.00 credit toward the purchase of any GIMIX system when you trade-in your working Color Computer, peripherals, and original software. The trade-in value is limited to 110% of the RADIO SHACK™ list price at the time your order is placed. You pay the freight. This offer is good only in the Continental U.S.; is limited to the first 100 orders; and expires on 9/30/84. Only one trade-in per customer.

GIMIX INC.
1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055

Now! All the "Stars" Together!



For Ordering Call TOLL FREE

1-800-338-6800

FROM – DATA-COMP, C.P.I.

C.P.I.
Color Micro Journal
'68' Micro Journal
Data-Comp
S.E. Media

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo
Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS+™

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.00 **Only $79.95**

STAR-DOS PLUS+
- Functions Same as FLEX
- Reads - writes FLEX Disks
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

$34.50

FLEX-CoCo Jr.
without TSC Editor & Assembler
$49.95

PLUS

TSC Editor
Reg $50.00
NOW $35.00

ALL VERSIONS OF FLEX & STAR-DOS• INCLUDE

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utilities
+ Super 800 Support
+ Free Color Micro Journal 1 yr. sub.
+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program
+ Many Many More!!!

TSC Assembler
Reg $50.00
NOW $35.00

DISK SYSTEMS FOR THE COLOR COMPUTER

THESE PACKAGES INCLUDE DRIVE, *CONTROLLER, POWER SUPPLY & CABINET, CABLE, AND MANUAL.

* SPECIFY WHAT CONTROLLER YOU WANT J&M, OR RADIO SHACK.

PAK #1 - 1 SINGLE SIDED, DOUBLE DENSITY SYS.	$389.95
PAK #2 - 2 SINGLE SIDED, DOUBLE DENSITY SYS.	$639.95
PAK #3 - 1 DOUBLE SIDED, DOUBLE DENSITY SYS.	$439.95
PAK #4 - 2 DOUBLE SIDED, DOUBLE DENSITY SYS.	$699.95
PAK #5 - 2 DOUBLE SIDED, DOUBLE DENSITY SYS. THINLINE DRIVES, HALF SIZE	$659.95
COLOR COMPUTER II 64K W/EXT. BASIC	$189.95

CONTROLLERS

J&M DISK CONTROLLER W/ JDOS OR RADIO SHACK DISK BASIC, SPECIFY WHAT DISK BASIC.	$139.95
RADIO SHACK DISK CONTROLLER 1.1	$134.95

DISK DRIVE CABLES

CABLE FOR ONE DRIVE	$ 19.95
CABLE FOR TWO DRIVES	$ 24.95

MISC

64K UPGRADE W/MOD. INSTRUCTIONS, C,D,E,F, AND COCO 2	$ 49.95
MJL KEYBOARDS	$ 69.95
MICRO TECH LOWER CASE ROM ADAPTER	$ 74.95
RADIO SHACK BASIC 1.2	$ 29.95
RADIO SHACK DISK BASIC 1.1	$ 29.95
RADIO SHACK EXT. BASIC	$ 39.95
SCREEN CLEAN CLEARS UP VIDEO DISTORTION	$ 39.95
MEMOREX DISKS 5" SS,DD	$ 24.00
SHIPPING INCLUDED ON DISK PRICES	
DISK DRIVE CABINET & POWER SUPPLY	$ 49.95
SINGLE SIDED, DOUBLE DENSITY 5" DISK DRIVE	$199.95
DOUBLE SIDED, DOUBLE DENSITY 5" DISK DRIVE	$249.95

PRINTERS

EPSON RX-80	$325.00
EPSON RX-80FT	$375.00
EPSON MX-100	$650.00
EPSON FX-100	$799.00
EPSON FX-80	$569.00
EPSON MX-70	$200.00

SERIAL BOARDS FOR PRINTERS

MX-SERIES	$119.95
FX-SERIES	$ 99.95

USA ADD 2% SHIPPING
FOREIGN ADD 5% SHIPPING

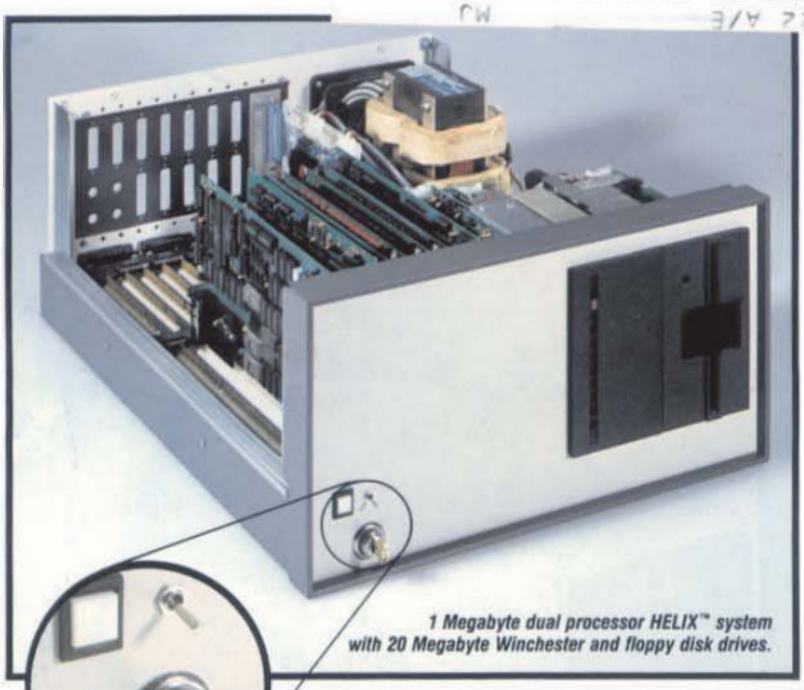
SPECIAL MX-100 $550.00

*FLEX is a Trademark of Technical System Consultants
*STAR-DOS+ is a Trademark of STAR-Kits & Data-Comp

5900 Cassandra Smith Rd. Hixson, TN 37343

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Chattanooga, TN
ISSN 0194-5025



1 Megabyte dual processor HELIX™ system with 20 Megabyte Winchester and floppy disk drives.

IN HAZELWOOD COMPUTER SYSTEMS demonstrates its leadership in computer technology by delivering the only computer system capable of switching between either the 6809 or the 68000 processor. Switching is easily accomplished by a simple front panel toggle switch. The reason we can offer this exclusive feature now, is that when our proven 6809 processor board was designed several years ago, we had the foresight to include the bus controls that allow processor switching.

Hazelwood Computer Systems is also proud to be the first S-50/S-64 bus manufacturer to license and deliver the OS9/68K Operating System from Microware Systems Corporation. OS9/68K is the 68000 version of the popular and powerful OS9 Operating System. Utilizing our proven MC-20 disk controller, OS9/68K can conveniently share a Winchester disk with OS9. Changing from 6809 to 68000 operation is as simple as switching processors and booting the new system from the Winchester disk.

The ease of switching processors and operating systems makes a HELIX™ dual processor system the natural choice for software development. In addition, the advanced design of HELIX™ equipment, emphasizing performance and reliability, makes HELIX™ boards and systems the best value in computing offered anywhere.

System prices vary with configuration. Call for exact pricing.

THE SWITCH IS ON...

The CP-08 processor board utilizes a 68008 processor running at 10 Mhz clock rate. Using proprietary bus synchronization circuitry and single cycle DMA, the CP-08 achieves a marked performance increase over a 2 MHz 6809. Offering absolute compatibility with the 68000 instruction set, the 68008 addresses up to 1 Megabyte of memory. Also included on the CP-08 are up to 4K of ROM, an interrupt timer, and, with battery backup operation, a clock/calendar and 2K RAM. Implemented as a standard S-50 board, the CP-08 brings 68000 operation to S-50 bus computers.

ORDER: CP-08 **PRICE: $595**

The MC-20 Mass Storage Controller board interfaces up to 4 floppy and 8 Winchester disk drives to the S-50/S-64 bus. The MC-20 is an intelligent controller with its own 2 Mhz 6809 processor and 56K RAM. It provides DMA data transfers to a full 24 bit address. All disk operation requests are by logical block number, with the controller performing the necessary track/sector address calculations. Any combination of 5¼ or 8 inch floppy drives can be accommodated with all drive parameters, such as write precompensation, software controlled for each individual drive. Winchester drives are connected via a SASI bus interface. Block address mapping is provided which allows a single drive to be segmented into several logical units. The MC-20 is the controller of the MS-20 Mass Storage Subsystem which includes a 20 Megabyte Winchester drive.

ORDER: MC-20 **PRICE: $695**

OS9/68K

OS9/68K offers increased performance and larger user memory space while retaining all of the features of OS9. Disk file compatibility and operational similarity assures that present OS9 users can easily transfer their operations to the 68000. Included are an editor, assembler, linker, and debugger. A C compiler is available now. BASIC09 and other languages will be avilable soon.

ORDER: OS9/68K **PRICE: $250**

All items available stock to 30 days.
Prices subject to change without notice.

HAZELWOOD COMPUTER SYSTEMS

907 East Terra, O'Fallon, MO 63366, 314-281-1055

HELIX™

OS9 and OS9/68K are registered trademarks of Microware Systems Corp. HELIX is a trademark of Hazelwood Computer Systems